



# آموزش Android Studio

مولف: مهندس افشن رفوآ

---

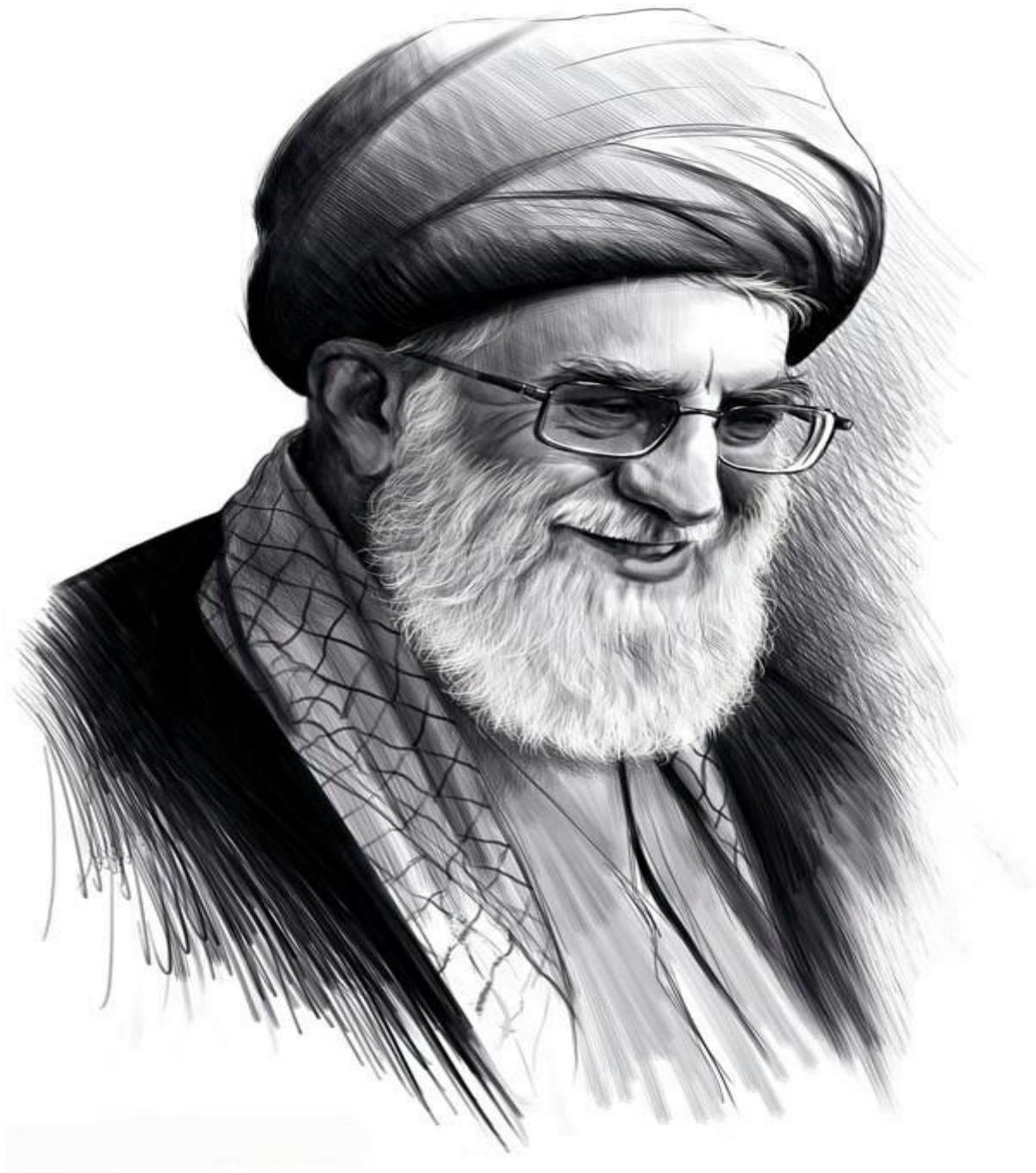
[www.tahlildadeh.com](http://www.tahlildadeh.com)



ب

آدرس آموزشگاه: تهران - خیابان شریعتی - بالاتر از خیابان ملک - جنب بانک صادرات - پلاک 651 طبقه دوم - واحد 7

88146323 - 88446780 - 88146330



تقدیم به تائب امام عصر، حضرت آیت الله خامنه‌ای که عصازدنش ضرب آهنگ حیدری دارد

ت

آدرس آموزشگاه : تهران - خیابان شریعتی - بالاتر از خیابان ملک - جنب بانک صادرات - پلاک 651 طبقه دوم - واحد 7

88146323 - 88446780 - 88146330



ث

آدرس آموزشگاه : تهران - خیابان شریعتی - بالاتر از خیابان ملک - جنب بانک صادرات - پلاک 651 طبقه دوم - واحد 7

88146323 - 88446780 - 88146330



67.....	1-5-2-نمونه ای از فایل تنظیمات اندروید (manifest)
68.....	1-5-3-خصیصه‌ی Package و تعین نسخه‌های اپلیکیشن با استفاده از خصیصه‌های Version در فایل تنظیمات (manifest)
69.....	1-5-4-تگ <application> (یک ظرف برای اجزا) و کامپوننت‌های دیگر
70.....	1-5-5-تعیین پایین/بالاترین نسخه‌ی اندروید که برنامه بر روی آن اجرا می‌شود (خصیصه‌های minSdkVersion و targetSdkVersion)
70.....	1-5-6-تعیین مجوز‌های دسترسی با استفاده از تگ permission
71.....	1-5-7-تعیین سیستم سخت افزاری مورد نیاز (بخش uses-feature و uses-configuration در فایل تنظیمات)
71.....	1-5-8-تعیین محل نصب (خصیصه‌ی installLocation)
72.....	1-6-1-فایل‌های محتوا (Resources)
74.....	1-6-2-فایل‌های resource (تعریف تعدادی ثابت رشته‌ای، آرایه‌ی رشته‌ای، ثابت رنگ و ابعاد) R.java
75.....	1-6-3-فایل‌های Layout (فایل‌های چیدمان رابط کاربری)
76.....	1-6-4-افزایش سرعت و کارایی با استفاده از layout های ساده
76.....	1-6-5-رہنمودها و روش‌های بهینه در خصوص ID های اختصاص داده شده به فایل‌های محتوا/resource
78.....	1-6-6-محتوای و منابعی که توسط سیستم اندروید ارائه می‌شود (system resources)
78.....	1-7-مفهوم view در اندروید - المان‌ها و ابزارک‌های رابط کاربری یا UI Widget
79.....	1-8-1-استفاده از layout manager
79.....	1-8-2- تنظیم و ویرایش layout manager ها از طریق attribute
81.....	1-8-3- Constraint Layout
82.....	1-8-4- FrameLayout
82.....	1-8-5- LinearLayout
84.....	1-8-6- RelativeLayout
84.....	1-8-7- GridLayout
86.....	1-8-8- ScrollView
88.....	1-8-9- افزودن radio button و radio group به قالب/layout
91.....	1-8-10- ویرایش orientation/وضعیت چیدمان radio group در زمان اجرا (به صورت dynamic)
92.....	1-8-11- تست اپلیکیشن
92.....	1-9-9- دسترسی به محتوای static و استفاده از آن
92.....	1-9-1- دسترسی به فایل‌های محتوا (resources) از طریق کلاس Resources
92.....	1-9-2- دسترسی به view ها از layout در کلاس activity

94.....	1-9-3-دسترسی به محتوای مورد نظر در فایل های XML از دیگر فایل های resource
95.....	1-9-4-استفاده از پوشه‌ی assets و دسترسی به داده‌های ذخیره شده در آن
96.....	1-9-5-افزودن عکس به اپلیکیشن
96.....	1-9-6-افزودن view جدید به پروژه
96.....	1-9-7-جایگزین کردن عکس‌ها با کلیک بر روی دکمه
97.....	1-9-8-تست اپلیکیشن
99.....	1-9-9-تست اپلیکیشن
100.....	1-9-10-ساخت پروژه
101.....	1-9-11-ایجاد attribute‌ها
102.....	1-9-12-ایجاد فایل Layout و تنظیم کننده‌ی ظاهر اپلیکیشن
106.....	1-9-13-ویرایش مقادیر property‌های view
108.....	1-9-14-معرفی یک کلاس کمکی (utility class)
109.....	1-9-15-ویرایش کد activity
110.....	1-9-16-راه اندازی اپلیکیشن
110.....	1-10-1-فرایند آماده سازی، عرضه، توزیع و نصب اپلیکیشن (Deployment)
110.....	1-10-1-نحوه‌ی آماده سازی و عرضه‌ی اپلیکیشن برای نصب
111.....	1-10-2-مشخص کردن اجزا نرم افزاری و سیستم سخت افزاری مورد نیاز برای اپلیکیشن
111.....	1-10-3-کردن (تخصیص امضای الکترونیکی جهت دیباگ) اپلیکیشن قبل از انتشار و عرضه‌ی آن در فروشگاه‌های مجازی
112.....	1-10-4-خروجی امضا شده دادن (Export) اپلیکیشن از طریق محیط کاری Android Studio
112.....	1-10-5-خروجی امضا شده دادن Export اپلیکیشن از طریق محیط برنامه نویسی (IDE) Eclipse
112.....	1-10-6-نصب اپلیکیشن از روش‌های دیگر
112.....	1-10-7-فروشگاه مجازی Google Play
113.....	1-11-1-آشنایی با محیط برنامه نویسی Android Studio
113.....	1-11-1-نوار ابزار محیط برنامه نویسی Android Studio
114.....	1-11-2- محل ذخیره سازی پروژه‌های اندروید
115.....	1-11-3-باز کردن پروژه‌ها و راه‌گزینی (سوییچ) بین آن‌ها
116.....	1-11-4-پاک کردن محتوای پوشه‌های build و کامپایل مجدد پروژه‌ها/همانگ سازی پروژه با Gradle
116.....	1-11-5-بروز رسانی تنظیمات Android Studio
117.....	1-11-6-استفاده از proxy با Android Studio
118.....	1-11-7-ابزار Android Device Monitor
118.....	1-12-شروع به استفاده از Android Studio

119 .....	12-1-ایجاد یک فایل layout جدید
120 .....	12-1-ایجاد یک فایل محتوای جدید (resource file)
121 .....	12-1-کار با فایل های layout
121 .....	12-1-قرار دادن فایل تصویری (image) در پروژه
123 .....	12-1-افزودن یک منوی جدید (menu resource)
125 .....	12-1-ایجاد فایل preference
127 .....	12-1-مشاهده محتوای فایل های log اپلیکیشن اندرویدی
127 .....	12-1-موظایف و task های مربوط به برنامه نویسی Java
127 .....	13-1-ایجاد کد Getter/Setter، ... toString
127 .....	13-2-استفاده از 8 Java در ساخت اپلیکیشن های اندرویدی
128 .....	13-3-استفاده از live template و تکه کدهای آماده در کد برنامه خود
128 .....	14-1-کار با نرم افزار کنترل نسخه Git
128 .....	14-1-کپی کردن یک Git repository یا Clone
129 .....	14-2-کپی کردن یک Github repository
129 .....	15-نظرارت بر مصرف حافظه (Memory Monitor)
129 .....	16-1-استفاده از Gradle در محیط برنامه نویسی IntelliJ
129 .....	16-1-محیط توسعه IntelliJ و قابلیت پشتیبانی از Gradle project view
130 .....	16-3-راه اندازی task های Gradle
130 .....	16-4-وارد کردن (import) یک پروژه ای از Gradle در محیط برنامه نویسی IntelliJ
133 .....	16-5-مشاهده نتایج اجرا شده
134 .....	17-1-انتقال پروژه از Eclipse به Android Studio
134 .....	17-1-ویرایش keybinding های Eclipse binding (تغییر تنظیمات صفحه کلید)
137 .....	1-آموزش Intent در Android
1-1-1-اجرا و راه اندازی دیگر کامپوننت ها و اجزا نرم افزاری اندروید به وسیله intent ها (تبادل اطلاعات بین اجزا و کامپوننت های مختلف به وسیله آبجکت های intent)	
137 .....	1-2-1-اندازی Activity ها به وسیله آبجکت intent
138 .....	1-2-2-راه اندازی service ها
139 .....	1-4-ارسال intent های صریح explicit و ضمنی implicit
140 .....	2-1-انقال و تبادل اطلاعات بین activity ها و service ها
141 .....	2-1-ارسال داده به کامپوننت مقصد
141 .....	2-2-بازیابی اطلاعات از یک subactivity

145 .....	2-ثبت و معرفی intent برای intent filter 3
145 .....	2-3-1 Intent filter چیست؟
146 .....	2-3-2 تعریف intent filter
147 .....	2-4-گوش فرا دادن به event ها (ارسال پیغام به سیستم اندروید از طریق intent)
148 .....	2-5-شناشایی intent receiver های مربوطه/بررسی اینکه آیا یک کامپونت به intent خاصی گوش فرا می دهد/برای آن ثبت شده یا خیر
149 .....	2-5-1 ساخت پروژه و فایل layout اصلی
149 .....	2-5-2 ایجاد یک فایل layout جدید
150 .....	2-5-3 ایجاد یک activity دیگر
151 .....	2-5-4 راه اندازی subactivity
152 .....	2-5-5-5 انتقال داده (ارسال مقدار) از activity اول به activity دوم (ResultActivity)
152 .....	2-5-6 دریافت اطلاعات ارسالی از activity اول (داده های intent) در ResultActivity
153 .....	2-5-7 دریافت و ارزیابی داده های بازگشتی (ارسالی از activity دوم) در کلاس MainActivity
156 .....	2-5-8 ایجاد پروژه
156 .....	2-5-9 ثبت و تخصیص یک activity به عنوان مرورگر
158 .....	2-5-10 تست برنامه
160 .....	2-5-11 تبدیل اپلیکیشن به یک مرورگر واقعی
167 .....	6-2-مدیریت چرخه حیات / application
167 .....	6-2-مدیریت چرخه حیات/ life cycle اپلیکیشن و activity در اندروید
169 .....	2-7-آجکت
171 .....	2-9-Activity/Lifecycle
171 .....	2-9-1 وضعیت های مختلف یک activity
174 .....	2-9-2 پیان یافتن activity ها توسط سیستم عامل
175 .....	2-10-داده ها و اطلاعات ذخیره شده از آجکت activity جهت بازگردانی به وضعیت قبلی/ state
177 .....	2-11-آجکت های جاوایی که در صورت تغییر در تنظیمات دستگاه باید بین نمونه های activity پاس داده شوند
178 .....	2-12-2-مدیریت configuration و تنظیمات
178 .....	2-12-1-جلوگیری از نابود و مجدد ایجاد شدن activity بر اثر تغییر در تنظیمات و وضعیت برنامه
179 .....	2-12-2-ثبت کردن وضعیت نمایش (orientation) یک activity (orientation)
181 .....	2-12-3-ایجاد activity های مورد نیاز
184 .....	2-12-4-بازگردانی وضعیت نمونه activity
187 .....	2-13-مبحث امنیت و مجوز های دسترسی در اندروید (Security&Permission)

187 .....	13-1-مفهوم امنیت در اندروید .....
187 .....	13-2-مفهوم مجوزهای دسترسی در اندروید .....
188 .....	13-3-سیستم مدیریت و تخصیص مجوز در اندروید پیش از API 23 .....
188 .....	13-4-سیستم تخصیص مجوز از API 23 به بعد (اعطای مجوز به هنگام اجرا برنامه) .....
191 .....	14-1-پیاده سازی list یا grid list یا جهت نمایش آیتم ها در اندروید .....
191 .....	14-2-پیاده سازی list (نمای فهرستی) و grid (نمای جدولی و خانه بندی شده همراه با سطر و ستون) در اپلیکیشن های اندروید به وسیله ای کلاس/ویجت RecyclerView .....
191 .....	14-3-استفاده از list یا grid در اپلیکیشن اندروید .....
193 .....	14-4-استفاده از RecyclerView جهت پیاده سازی لیست های پیچیده .....
194 .....	14-5-استفاده از Adapter ها .....
197 .....	14-6-Layout manager پیش فرض .....
197 .....	14-7-کلاس های مورد نیاز برای پیاده سازی RecyclerView .....
199 .....	14-8-مدیریت event های مربوط به کلیک یا لمس نمایشگر در لیست/recycler view .....
200 .....	14-9-استفاده از layout های مختلف در recycler view .....
201 .....	14-10-پیاده سازی اینیمیشن های اختصاصی .....
201 .....	14-11-فیلتر و مرتب سازی داده ها .....
201 .....	14-12-بروز رسانی داده ها در کلاس adapter .....
202 .....	14-13-ایجاد پروژه ها و افزودن dependency مورد نظر در فایل gradle build .....
203 .....	14-14-ایجاد فایل های layout مورد نیاز .....
206 .....	14-15-استفاده از کتابخانه data binding جهت اتصال دو طرفه داده ها از Model به UI در اندروید .....
208 .....	14-16-فایل های layout اتصال داده از model به UI (Data Binding Layout Files) .....
210 .....	14-17-data binding و مدیریت event ها .....
210 .....	14-18-بروز رسانی رابط کاربری با تغییر رخداده در data model .....
211 .....	14-19-پیاده سازی data binding و نمونه ای از کاربرد آن .....
213 .....	14-20-افزودن کتابخانه data binding به پروژه .....
213 .....	14-21-ساخت کلاس جهت مطلع ساختن view از تغییرات در model .....
214 .....	14-22-تنظیم و ویرایش فایل ها برای استفاده از قابلیت data binding .....
216 .....	14-23-استفاده از کلاس ListView در اندروید .....
216 .....	14-24-پیاده سازی list در اندروید با کلاس ListView .....
218 .....	14-25-اندروید و/manan رابط کاربری widget .....
218 .....	14-26-View هایی برای مدیریت لیست ها .....
219 .....	14-27-نوع داده ای آیتم های لیست .....

219 .....	..... Adapter-2-17-3
221 .....	..... 2-فیلتر و مرتب سازی داده ها ..... 17-4
221 .....	..... 2-بروز رسانی داده ها در adapter ..... 17-5
221 .....	..... 2-معرفی listener برای آگاهی از تغییرات ..... 17-6
221 .....	..... Adapter پیش فرض ..... 2-18
222 .....	..... 2-کلاس های adapter پیش فرض محیط (platform) اندروید ..... 18-1
222 .....	..... 2-استفاده از ArrayAdapter ..... 18-2
223 .....	..... 2-نمونه ای استفاده از ListView با ArrayAdapter ..... 18-3
224 .....	..... 2-پیاده سازی adapter های اختصاصی ..... 19
224 .....	..... 2-ساخت یک adapter اختصاصی ..... 19-1
225 .....	..... 2-آماده سازی سطر از لیست ..... 19-2
227 .....	..... 2-بروز رسانی data model از طریق کلاس adapter ..... 19-3
227 .....	..... ListFragment و ListActivity-2-20
228 .....	..... 2-ظرف آماده و پیش فرض برای استفاده از ListView ..... 20-1
229 .....	..... 2-layout و ListActivity-2-20-2
229 .....	..... 2-انتخاب یک view مکان نگهدار/placeholder برای لیست خالی ..... 20-3
235 .....	..... 2-ListView و مبحث کارایی و سرعت اپلیکیشن ..... 21
236 .....	..... 2-عملیات سنگین و زمان بر ..... 21-1
236 .....	..... 2-جلوگیری از inflation فایل های layout و ایجاد آبجکت های جاوا جهت بهینه سازی برنامه ..... 21-2
237 .....	..... 2-استفاده از الگوی توسعه View holder در اپلیکیشن جهت بهینه سازی ..... 21-3
239 .....	..... 2-ذخیره سازی انتخاب یک view (تعیین وضعیت انتخاب با متد (setChoiceMode()
241 .....	..... 2-استفاده از Contextual action mode ListView برای actionbar موقتی که بر روی ..... 23
241 .....	..... قرار می گیرد ..... 23
243 .....	..... 2-پیاده سازی قابلیت لغو undo action ..... 24
244 .....	..... 2-چه زمانی باید قابلیت لغو عملیات را به برنامه اضافه کنید ..... 24-1
248 .....	..... 2-بهینه سازی کارایی و سرعت اپلیکیشن ..... 25
249 .....	..... 2-آموزش نحوه نمایش دو آیتم در یک ListView ..... 26
250 .....	..... 2-انتخاب چندین آیتم در ListView ..... 27
250 .....	..... 2-تعامل بین ListView و model ..... 27-1
255 .....	..... 2-پیاده سازی یک ListView کشویی و قابل بسط ..... 28
255 .....	..... ExpandableListView-2-28-1
257 .....	..... 2-پیاده سازی یک ExpandableListView ..... 28-2

261.....	2-آموزش متفرقه
261.....	2-29-2-معرفی کردن یک گوش فراخوان به کلیک طولانی مدت (LongItemClickListener) به آیتم های لیست .....
261.....	2-29-2-اضافه کردن Header و Footer به لیست.....
262.....	SimpleCursorAdapter-2-30
263.....	2-31-سایر کتابخانه های کد باز (Open Source libraries)
264.....	2-32-ثبت وقایع و گزارش گیری (Logging) در اندروید.....
265.....	ثبت وقایع و گزارش گیری با استفاده از ابزار LogCat Android logging/LogCat
265.....	2-33-1-سیستم گزارش گیری اندروید (log system)
265.....	2-33-2-ایجاد دستورات گزارش گیری (log statement)
266.....	2-34-2-پیاده سازی dialog در اپلیکیشن به وسیله ی DialogFragments
267.....	نمایش پنجره ی محاوره dialog به وسیله ی DialogFragments
267.....	2-34-1-به نمایش گذاشتن dialog در برنامه .....
267.....	2-34-2-استفاده از dialog های آماده و از قبل موجود .....
268.....	2-34-3-معرفی layout اختصاصی برای DialogFragment
268.....	2-34-4-تعامل activity با DialogFragment .....
269.....	2-34-5-ایجاد پروژه و فایل های layout .....
269.....	2-34-6-معرفی activity و تنظیم fragment .....
274.....	طراحی صفحات تک قطعه و چند قطعه (multi-pane & single-pane) با استفاده از fragment ها در اندروید/ایجاد ایاهای انعطاف پذیر به وسیله ی fragment ها .....
274.....	3-1-1-شرح Layout های تک قطعه (single-pane) و چند قطعه (multi-pane) .....
277.....	3-1-2-شرح مفهوم Fragment .....
278.....	3-1-3-Fragment ها و دسترسی به Context .....
278.....	3-1-4-مزایای استفاده از fragment .....
281.....	3-1-5-طراحی اپلیکیشنی با UI انعطاف پذیر و سازگار با نمایشگرهای مختلف به وسیله ی fragment ها .....
282.....	3-2-2-معرفی و استفاده از fragment ها .....
282.....	3-2-1-تعريف fragment .....
283.....	3-2-2-معامل اپلیکیشن با fragment ها .....
284.....	3-2-3-ارسال پارامتر به fragment ها .....
285.....	3-3-چرخه ی حیات (life cycle) fragment .....
288.....	3-4-معرفی activity برای fragment .....

289 .....	3-4-1 تعریف fragment برای activity خود در فایل layout به صورت static
289 .....	3-4-2 مدیریت fragment در زمان اجرا و به صورت dynamic
291 .....	3-4-3 بررسی اینکه آیا یک fragment در layout حاضر است یا خیر
291 .....	3-4-4 بررسی تعداد fragment ها
292 .....	3-4-5 افزودن آجکت backstack به fragment transition
292 .....	3-4-6 استفاده از افکت های تعریف شده توسط Property Animation API برای حرکت بین fragment ها
293 .....	3-4-7 ماندگارسازی و ذخیره ی دائمی داده در fragment ها
293 .....	3-4-8 نگهداری و بازگردانی اطلاعات پس از تغییر در تنظیمات config
293 .....	3-5 Fragment ها و پردازش در پس زمینه (background processing)
293 .....	3-5-1 Fragment های فاقد UI
294 .....	3-5-2 Retained headless fragment (config changes) ها جهت مدیریت تغییر در تنظیمات
296 .....	3-5-3 تعریف فایل های layout برای fragment ها
297 .....	3-5-4 ایجاد کلاس های fragment
298 .....	3-5-5 ویرایش فایل layout اصلی
299 .....	3-5-6 RssfeedActivity کلاس
301 .....	3-5-7 تعریف activity layout ویژه ی نمای عمودی
301 .....	3-5-8 تعریف یک flag یا گزینه ی بولی مستقل از resource selector (گزینشگر منابع)
302 .....	3-5-9 تنظیم و ویرایش کلاس RssfeedActivity
305 .....	3-5-10 ذخیره ی آخرین مقدار انتخابی در یک headless fragment
306 .....	3-5-11 تعریف یک flag یا گزینه ی بولی مستقل از گزینشگر resource
306 .....	3-5-12 تنظیم و ویرایش کلاس RssfeedActivity
309 .....	استفاده از نوارابزار در اپلیکیشن های اندروید Action bar در اندروید
309 .....	3-6 Toolbar شرح مفهوم
310 .....	3-6-1 Toolbar در اندروید چیست و چه کاربردی دارد؟ (action bar)
API بر روی دستگاه هایی که ورژن کتابخانه های اندروید مورد استفاده در آن ها پایین تر از 21 است (21)	3-6-2 Actionbar
311 .....	3-6-3 Options menu
311 .....	3-7 استفاده از Toolbar
312 .....	3-7-1 تعریف آیتم های نوار ابزار toolbar actions در
313 .....	3-7-2 عملیاتی که در پی انتخاب آیتم های نوارابزار رخ می دهد (واکنش نشان دادن به انتخاب action ها)
314 .....	3-7-3 جستجو برای یک action یا آیتم در نوارابزار action bar
314 .....	3-7-4 ویرایش منو

314.....	contextual action (وایسته کردن toolbar با استفاده از toolbar و Contextual action mode-3-7-5 mode)
315.....	-3-اضافه نمودن آیتم به action bar با استفاده از fragment ها
315.....	3-7-7- تنظیم قابلیت رویت (visibility) و دسترسی toolbar
315.....	3-7-8- تخصیص یک عکس drawable به action bar
316.....	3-7-9- پنهان یا کم رنگ کردن دکمه ی پیمایش (diming navigation button)
318.....	3-7-10- استفاده از حالت نمایش تمام صفحه (immersive full screen mode)
319.....	3-7-11- پیاده سازی قابلیت دو نیم کردن نوار ابزار (split toolbar)
320.....	3-7-12- ایجاد محتوا و منابع مربوطه res/menu در پوشه ی menu resource
320.....	3-7-13- تنظیم و ویرایش کد برنامه Dynamic-3-8
324.....	View-3-8-1- هایی با پیاده سازی اختصاصی در action bar
325.....	Action view-3-8-2
327.....	Action provider-3-9
327.....	3-9-1- Action provider چیست و چه کاربردی دارد؟
329.....	3-10- پیمایش در اپلیکیشن از طریق آیکون
329.....	3-10-1- استفاده از آیکون اپلیکیشن در action bar جهت راهبری به صفحه ی اصلی
331.....	3-10-2- استفاده از آیکون اپلیکیشن به عنوان دکمه ی Up
332.....	3-10-3- اضافه کردن منابع لازم برای منو
333.....	3-10-4- غیرفعال کردن action bar پیش فرض از طریق تگ style
333.....	3-10-5- اضافه کردن یک آجکت (view) به toolbar layout activity
336.....	3-10-6- حذف داده های اولیه
336.....	3-10-7- بروز رسانی myListFragment، در صورتی که refresh انتخاب شد
337.....	3-11- پیاده سازی الگو توسعه و قابلیت Swipe-To-Refresh (کشیدن صفحه به پایین جهت بروز رسانی لیست) در اپلیکیشن
341.....	3-11- دیتابیس SQLite و SQLite-4-1
341.....	4-1-1- SQLite چیست و چه کاربردی دارد؟
342.....	4-1-2- در اندروید
342.....	4-2- معماری SQLite
342.....	4-2-1- Package ها
343.....	4-2-2- ایجاد و آپدیت دیتابیس با SQLiteOpenHelper

344 .....	4-2-3-کلاس پدر SQLiteDatabase
346 .....	4-2-4-آجکت Cursor
347 .....	4-2-5-ListView، SimpleCursorAdapter و ListActivity ها
347 .....	4-3-آموزش: استفاده از SQLite
348 .....	4-3-1-مقدمه ای بر پروژه
349 .....	4-3-2-Database و Data Model ساخت
352 .....	4-3-3-UI و ظاهر ساخت
354 .....	4-3-4-اپلیکیشن و اجرای نصب
354 .....	4-4-4-شرح مفهوم Content Provider
354 .....	4-4-1-Content Provider چیست و چه کاربردی دارد؟
355 .....	4-4-2-URI و تعریف قالب پایه ای
356 .....	4-4-3-دسترسی به content provider
356 .....	4-4-4-پیاده سازی content provider اختصاصی
357 .....	4-4-5-Content provider و مبحث امنیت
358 .....	4-4-6-Thread safety (synchronized) مشکل همزمانی با استفاده از کلیدواژه ای
359 .....	4-4-6-ایجاد contact ها (مخاطبین) در محیط شبیه ساز
361 .....	4-4-7-استفاده از Contact Content Provider
363 .....	4-5-Loader و Cursor ها
366 .....	4-5-1-Database کلاس های مدیریت
368 .....	4-5-2-ContentProvider ایجاد
372 .....	4-5-3-Resource (ها) منابع
373 .....	4-5-4-layout فایل های تعریف
376 .....	4-5-5-Activity ها تنظیم
382 .....	4-5-6-SQLite محل ذخیره سازی دیتابیس
383 .....	4-5-7-Shell access (Parser) command line دسترسی به دیتابیس از راه دور به وسیله ای
386 .....	4-6-xmlPullParser (Parser) پردازش فایل های XML در اندروید با استفاده از تحلیلگر نحوی
386 .....	4-6-XmlPullParser تحلیلگر نحوی فایل های XML با استفاده از
389 .....	4-6-File API (preferences API) ذخیره ای ماندگار داده ها در اپلیکیشن اندروید با استفاده از
389 .....	4-7-File based persistence (ذخیره داده ها در سیستم فایل)
389 .....	4-7-1-(local data persistence) روش های ذخیره ای ماندگار داده ها به صورت محلی
391 .....	4-7-2-ذخیره سازی internal داده در مقایسه با ذخیره ای داده ها به صورت external

391.....	4-7-3-جایگذاری اپلیکیشن در حافظه ی خارجی.....
391.....	4-8-Preferences-(ذخیره و بازگردانی اطلاعات مربوط به تنظیمات کاربر)
391.....	4-8-1-ذخیره جفت های کلید-مقدار از نوع داده ای اولیه .....
393.....	4-8-2-گوش فرآخوانی به تغییرات در تنظیمات کاربر به وسیله ی preference listener .....
394.....	4-8-3-ایجاد فایل preference .....
394.....	4-8-4-ایجاد activity جهت دریافت ورودی از کاربر .....
394.....	4-8-5-متصل کردن activity مربوط به تنظیمات و دریافت ورودی کاربر .....
395.....	4-8-6-بارگذاری RSS feed با استفاده از مقدار preference .....
397.....	4-9-File API.....
397.....	4-9-1-استفاده از File API .....
399.....	4-9-2-حافظه و محل ذخیره سازی خارجی (external storage) .....
401.....	4-10-اتصال به اینترنت، اجرای عملیات HTTP و دسترسی به منابع موجود در سطح وب در اندروید/Android networking .....
401.....	4-10-1-مروری بر اتصال به اینترنت و دسترسی به منابع از اینترنت در اندروید .....
402.....	4-10-2-مجوز اتصال به اینترنت .....
402.....	4-10-3-بررسی وضعیت اتصال به اینترنت .....
402.....	4-10-4-روش های بهینه برای اتصال و دسترسی به اینترنت در اندروید .....
404.....	4-11-استفاده از کتابخانه ی Retrofit 2.0 به عنوان REST Client .....
406.....	4-11-1-ساخت پروژه و تنظیمات اولیه .....
407.....	4-11-2-معرفی API و کلاس Retrofit adapter .....
413.....	4-12-آموزش RxJava 2.0 .....
413.....	4-12-1-RxJava و شرح مفهوم برنامه نویسی Reactive یا ناهزمان .....
414.....	4-12-2-اضافه کردن کتابخانه 2.0 RxJava .....
415.....	4-12-3-برنامه نویسی ناهزمان (Async) .....
416.....	4-12-4-Observable ها و Observer ها .....
416.....	4-12-5-معرفی thread اجرا و یک thread برای گوش دادن به تغییرات (observe) .....
416.....	4-13-Operator ها .....
417.....	4-14-استفاده از delay و به تعویق انداختن تولید و ارسال نتیجه .....
417.....	4-14-1-ایجاد Observable ها و Observer ها .....
418.....	4-14-2-انجام عملیات تبدیل .....
419.....	4-14-2-Subject ها .....
419.....	4-15-آجکت Single یا همان Promise (مکان نگهدار مقدار مورد نظر) .....

4-دور انداختن subscription ها (قطع اتصال بین observable و observer) و استفاده از کلاس CompositeDisposable	421
4-ذخیره ی موقتی مقدار Observable های ارسال شده (completed observables)	422
4-Backpressure و Flowable<T>	424
4-تبديل نوعی به نوع دیگر	424
4-تست Observable و Subscription های RxJava	426
4-تست observable ها	426
آموزش پیاده سازی پردازش پس زمینه ای و ناهمگام با Loader ، Handler و AsyncTask	427
4-پردازش پس زمینه ای در اندروید	427
4-چرا استفاده از مفهوم همووندی concurrency توصیه می شود؟	427
4-Thread اصلی یا UI thread	428
4-پردازش ناهمزن و استفاده از thread ها در اندروید	429
4-ارائه ی feedback و توضیح مختصر برای کاربر در طول اجرای عملیات طولانی	429
4-کلاس Handler	430
4-استفاده از کلاس Handler جهت مدیریت همووندی	430
4-ایجاد و استفاده ای مجدد از thread اصلی	430
4-کلاس AsyncTask	433
4-هدف استفاده از کلاس AsyncTask	433
4-استفاده ای کاربردی از کلاس AsyncTask	433
4-اجرای همزمان چندین AsyncTasks	434
4-پردازش پس زمینه ای و مدیریت چرخه ی حیات (lifecycle)	436
4-حفظ اطلاعات مربوط به وضعیت (state) بین تغییراتی که در نحوه ی پیکربندی رخ می دهد	436
4-استفاده از آجکت application جهت ذخیره اطلاعات چندین آجکت	437
4-Fragment ها و پردازش موازی در پس زمینه (background processing)	439
4-نگهداری اطلاعات آجکت جهت بازگردانی آن پس از تغییر در نحوه ی پیکربندی	439
4-UI های فاقد Fragment (headless fragments)	439
4-کلاس Loader	439
4-هدف از کاربرد کلاس Loader	440
4-پیاده سازی کلاس Loader	440
4-دبتابیس SQLite و پیاده سازی کلاس CursorLoader	441
4-پیاده سازی کلاس service	442
4-استفاده از service ها	444

449 .....	JSON و اندروید ..... 4-26
450 .....	4-26-1-کتابخانه ی پیش فرض و درون ساخته ی اندروید برای پردازش JSON
450 .....	4-26-2-ساخت JSON .....
453 .....	بخش اول : ..... 453 ..... استفاده از drag & drop در اندروید ..... 5-1
453 .....	5-1-1-پیاده سازی قابلیت کشیدن view ..... 453
454 .....	5-1-2-معرفی مشخصات و اطلاعات محل جایگذاری view (مشخص کردن drop target view) ..... 454
456 .....	5-1-3-ایجاد فایل های Drawable (فایل های تصویری تعریف شده در فرمت XML) ..... 456
457 .....	5-1-4-Activity و تنظیم فایل layout مربوطه ..... 457
461 .....	بخش دوم : ..... 461 ..... Drawable چیست? ..... 5-2
462 .....	5-3-استفاده از drawable ها در view ها ..... 462
463 .....	5-4-بارگذاری Bitmap ها و drawable ها ..... 463
464 .....	5-5-Drawable های مبتنی بر XML ..... 464
464 .....	5-5-1-Shape ..... 464
466 .....	5-5-2-Drawable های مبتنی بر state ..... 466
466 .....	5-5-3-Drawable هایی که طی انتقال جایگزین drawable دیگری می شوند (transition drawable) ..... 466
467 .....	5-5-6-Drawable های برداری/توسعه پذیر بدون از دست رفت کیفیت (vector drawable) ..... 467
468 .....	5-5-7-Drawable animation (تعریف انیمیشن با بارگذاری یک drawable پس از دیگری) ..... 468
468 .....	5-7-1-Drawable nine-patch های (فایل های ترسیم شونده ی منعطف با کناره های بسط پذیر) ..... 468
469 .....	5-8-Drawable های اختصاصی ..... 469
470 .....	5-9-ساخت drawable های اختصاصی ..... 470
472 .....	بخش سوم : ..... 472 .....
472 .....	5-10-اصول طراحی UI در اندروید ..... 472
475 .....	5-10-1-طراحی انعطاف پذیر و واکنش گرا برای اپلیکیشن (Responsive design) ..... 475
477 .....	5-11-استفاده از style و theme در اپلیکیشن ..... 477
477 .....	5-11-1-طراحی ساده و بهینه برای اپلیکیشن های اندرویدی با material design/تم گذاری و ..... 477
478 .....	5-11-2-شرح مفهوم style و theme ..... 478
479 .....	5-12-3-دسترسی و اشاره به attribute ها در theme ..... 479
481 .....	5-12-4-Theme چیست? ..... 481
482 .....	5-13-استفاده از theme های خود محیط اندروید ..... 482
482 .....	5-13-1-استفاده از material design و کتابخانه ی لازم برای پشتیبانی از آن در طراحی اپلیکیشن اندرویدی ..... 482

483 .....	13-2-تنظیم رنگ های پایه ی theme (Styling the color palette) .....
484 .....	13-3-سبک دهی و تنظیم ظاهر view group های فردی و view .....
484 .....	14-استفاده از theme از پیش تعریف شده و آماده .....
484 .....	14-تمرین: پیاده سازی یک theme اختصاصی .....
486 .....	14-بخش چهارم : .....
486 .....	15-مرور کلی .....
487 .....	15-1-Live Wallpapers / تصاویر زنده .....
487 .....	15-2-live wallpaper .....
488 .....	15-3-wallpaper برای تنظیم intent .....
495 .....	15-بخش پنجم : .....
496 .....	16-شرح مفهوم Widget در اندروید .....
496 .....	16-1-AppWidgets .....
496 .....	16-1-1-مراحل ساخت یک widget .....
496 .....	16-2-widget .....
497 .....	16-3-andازه ی widget .....
497 .....	17-ایجاد Broadcast receiver برای widget .....
497 .....	17-1.widget .....
498 .....	17-2-view ها و layout های موجود و قابل استفاده .....
499 .....	17-AppWidgetProvider .....
500 .....	17-3-Receiver و پردازش ناهمزن .....
500 .....	18-بروز رسانی widget .....
507 .....	19-Collection View Widget .....
509 .....	20-فعال سازی یک widget برای قفل نمایشگر دستگاه (lock screen) .....
513 .....	21-بخش ششم : .....
513 .....	21-View های اختصاصی .....
513 .....	21-1-View های پیش فرض محیط اندروید .....
515 .....	21-2-اندروید چگونه view hierarchy را ترسیم می نماید! .....
516 .....	21-3-استفاده از view های جدید در فایل های layout .....
516 .....	21-4-تهیه ی تصویر آنی (screenshot) از view ها .....
517 .....	22-view های ترکیبی (Compound views) .....
518 .....	23-ساخت view های اختصاصی .....
518 .....	23-1-پیاده سازی view های اختصاصی .....
518 .....	23-2-اندازه گیری view ها .....

519 .....	5-23-3 تعریف layout manager اختصاصی
519 .....	5-24 Life Cycle
519 .....	5-24-1 event
520 .....	5-24-1 Event های مربوط به life cycle (Traversal life cycle event) به صورت ترتیبی/پیمایشی
521 .....	5-24-2 چرخه ای حیات
521 .....	5-25 تعریف های بیشتر برای view های اختصاصی
524 .....	5-25-1 تعریف و استفاده از attribute های جدید
525 .....	5-25-2 ساخت View ترکیبی
526 .....	5-25-3 تنظیم و ویرایش activity
528 .....	5-26 Canvas API (توابع مرتبط با کلاس Canvas)
528 .....	5-26-1 شرح مفهوم Canvas API
529 .....	5-26-1 کلاس Canvas
529 .....	5-26-2 کلاس Paint
530 .....	5-26-3 Shader
530 .....	5-26-4 ذخیره ای دائمی و ماندگارسازی داده های view
531 .....	بخش هفتم :
532 .....	5-27 تنظیمات و پیکربندی های مختلف دستگاه های اندروید
532 .....	5-27-1 طراحی برنامه برای انواع دستگاه های اندروید (با نمایشگر و اندازه های مختلف)
532 .....	5-27-2 Resource qualifier (تعریف کننده و گزینشگر منابع)
533 .....	5-27-3 Resource های مهم
533 .....	استفاده از وضعیت نمایش/orientation به عنوان یک resource qualifier (گزینش منابع بر اساس وضعیت نمایش) ....
533 .....	5-27-4 Version qualifier
533 .....	5-27-5 Width&Height به عنوان گزینشگر منبع
534 .....	5-28 String ها و محتوای متغیر (ترجمه ای نوشته های برنامه)
535 .....	5-28-1 Plural
536 .....	5-28-2 استفاده از google translate
536 .....	5-28-3 مدیریت مبحث کیفیت تصویر و تراکم پیکسلی نمایشگر های مختلف
538 .....	5-28-4 استفاده از density به عنوان گزینش گر منابع (انتخاب محتوا بر اساس چگالی پیکسلی)
539 .....	5-28-5 ارائه ای آیکون در اندازه های مختلف
540 .....	5-29 تعیین اندازه ای کامپوننت های UI در فایل های layout
540 .....	5-29-1 اندازه بندی در ابعاد ثابت (fixed) یا نسبی (relative)
540 .....	5-29-2 استفاده از واحد dp برای اندازه بندی و تعیین ابعاد المان های UI به صورت نسبی

541 .....	5-اندازه بندی نوشته ها بر حسب واحد sp
541 .....	5-تعیین اندازه ی کامپوننت های UI در کد برنامه (source code)
542 .....	بخش هشتم :
543 .....	5-انیمیشن سازی در اندروید
543 .....	5-استفاده از انیمیشن در اندروید
543 .....	5-کلاس های AnimatorListener و Animator
544 .....	5-کلاس ViewPropertyAnimator
546 .....	5-Layout animations
546 .....	5-اعمال انیمیشن بر روی انتقال بین activity ها (پیاده سازی انیمیشن بر روی transition بین دو activity)
552 .....	5-اعمال انیمیشن بر روی انتقال بین activity ها در اندروید با shared view ها
557 .....	6-بخش اول :
557 .....	6-1-سرویس های اندروید
557 .....	6-1-1-سرویس چیست
558 .....	6-1-2-سرویس ها و پردازش پس زمینه ای (background processing)
558 .....	6-1-3-سرویس های خود محیط اندروید (platform) و سرویس های اختصاصی
559 .....	6-1-4-راه اندازی و تعریف سرویس های اختصاصی
559 .....	6-1-5-سرویس های پیش زمینه (foreground)
559 .....	6-2-تعریف سرویس های اختصاصی
560 .....	6-2-1-پیاده سازی و اعلان
561 .....	6-2-2-فرایند راه اندازی و اجرای سرویس
562 .....	6-2-3-رفتار بازآغازی و restart سرویس
563 .....	6-2-4-متوقف کردن یک سرویس
564 .....	6-3-منتصل کردن دو طرفه ی سرویس ها (service binding)
564 .....	6-3-1-وصل شدن از activity ها به سرویس ها
565 .....	6-3-2-اتصال به سرویس های محلی
565 .....	6-3-3-اتصال به سرویس با استفاده از IPC
565 .....	6-4-اجرای سرویس ها در فرایندهای محزا
565 .....	6-4-1-اجرای یک سرویس در فرایند مختص به خود
566 .....	6-4-2-چه زمان می بایست یک سرویس را در فرایند محزا اجرا کرد؟
567 .....	6-4-3-تبادل داده و ارتباط با سرویس ها
567 .....	6-4-4-1-روش های مختلف برای برقراری ارتباط با سرویس ها
567 .....	6-4-4-2-استفاده از داده های کپسوله شده در intent

567 .....	6-استفاده از receiver
568 .....	6-اتصال به سرویس محلي activity
569 .....	6-Messenger پا ResultReceiver و Handler
569 .....	6-اتصال به سرویس در فرایند دیگر با استفاده از AIDL
580 .....	بخش دوم :
580 .....	6-5-زمان بندی تسک ها
581 .....	6-5-نحوه ی پیاده سازی
581 .....	6-5-2-روش ها و ابزار مختلف زمان بندی تسک ها
581 .....	6-6-زمان بندی background task ها با استفاده از JobScheduler
582 .....	6-6-1-توابع کتابخانه ای job scheduler
582 .....	6-6-2-مزایای استفاده از JobScheduler
583 .....	6-6-3-نحوه ی ایجاد یک Job
584 .....	6-6-4-ایجاد JobService
585 .....	6-6-5-ساخت activity آزمایشی
586 .....	6-6-6-ایجاد receiver
587 .....	6-6-7-ایجاد Job
588 .....	بخش سوم :
588 .....	6-7-Broadcast receiver
588 .....	6-7-1-شرح مفهوم broadcast receiver
589 .....	6-7-2-پیاده سازی receiver
589 .....	6-7-3-Lifecycle یا چرخه ی حیات کامپونت broadcast receiver
589 .....	6-7-4-پردازش ناهمزمان (async processing)
590 .....	6-7-5-محدودیت هایی در تعریف و استفاده از broadcast receiver
590 .....	6-7-6-ارسال broadcast receiver به اپلیکیشن به منظور تست
591 .....	6-7-7-شرح مفهوم Pending Intent
592 .....	6-8-Broadcast ها و رخدادهای سیستمی
592 .....	6-9-اجرا و راه اندازی سرویس به صورت خودکار از Receiver
592 .....	6-9-1-پیاده سازی receiver برای گوش فرآخوانی به رخدادهای مربوطه به وضعیت تلفن و تماس ها
595 .....	6-9-2-اعطا مجوز گوش فرآخوانی به تغییرات مربوط به وضعیت در receiver
600 .....	6-10-تعريف broadcast receiver به صورت dynamic (در زمان اجرای برنامه و به وسیله ی کدهای جاوا)
600 .....	6-10-1-Receiver هایی که به صورت dynamic اعلام شده
601 .....	6-10-2-استفاده از package manager جهت غیرفعال سازی static receiver های

601 .....	Intent-6-10-3 های ماندگار (Sticky broadcast intent)
602 .....	بخش چهارم : ..... ..... بخش چهارم
603 .....	Notification manager-6-11 ..... Notification manager
603 .....	6-11-1-شرح مفهوم notification manager
604 .....	6-11-2-تنظیم و مقدار دهی notification ها
607 .....	6-11-3-لغو کردن notification ها
609 .....	بخش پنجم : ..... ..... در اندروید Backup-6-12
610 .....	6-12-1-هدف از تهیهٔ نسخهٔ پشتیبان از دادهٔ داده
611 .....	6-12-2-تهیهٔ نسخهٔ پشتیبان از shared preferences (دادهٔ کوچک همچون اطلاعات مربوط به تنظیمات اپلیکیشن) و فایل ها
611 .....	6-12-3-تهیهٔ Backup کلی
613 .....	6-12-4-راه اندازی و فعال کردن پروسه های backup و restore (بازگردانی داده ها)
614 .....	بخش ششم : ..... ..... 6-هنمودهایی جهت طراحی و ارائهٔ اپلیکیشن های کارامد
614 .....	6-13-1-چرا باید در استفاده از منابع اندروید مراقب بود؟
614 .....	6-13-2-اجتناب از تخصیص و ایجاد آجکت غیر ضروری
615 .....	6-13-3-استفاده از data structure های کارامد
616 .....	6-14-مدیریت bitmap
618 .....	6-15-استفاده از Cache
618 .....	6-15-1-استفاده از حافظهٔ نهان
620 .....	6-15-2-پاک سازی cache
622 .....	بخش اول : ..... ..... 7-هدف از نوشتن تست های نرم افزاری چیست؟
622 .....	7-1-7-1-شرح مفهوم تست های نرم افزاری
622 .....	7-1-2-چرا استفاده از تست های توصیه می شود؟
623 .....	7-1-3-فریم ورک های تست گیری (testing frameworks) برای Java
623 .....	7-2-1-واژه ها و مفاهیم مرتبط با تست گیری
623 .....	7-2-1-کد (یا اپلیکیشن) مورد تست / Code Under Test
624 .....	7-2-2-Test fixture (مقادیر ثابت تست)
624 .....	7-2-3-شرح مفهوم Unit test و تست بخش های مختلف نرم افزار (کد) به صورت مجزا
625 .....	7-2-4-Integration test (تست اپلیکیشن در context و بستر خودش)

625 .....	تست های بررسی کارایی (Performance test-7-2-5)
625 .....	State testing و Behavior testing-7-2-6
626 .....	7-3-سازماندهی تست
626 .....	7-3-1-تست در کجا بایستی نوشته شود
626 .....	7-3-2-کدام بخش نرم افزار بایستی تست شود
627 .....	7-4-استفاده از JUnit
627 .....	7-4-1-(فریم ورک unit test بر روی پروژه های جاوا) JUnit framework
628 .....	7-4-2-نحوه ای طراحی یک تست در فریم ورک تست گیری JUnit؟
628 .....	7-4-3-نمونه ای از JUnit test
629 .....	7-4-4-4-قرارداد/روش های نام گذاری تست های مبتنی بر JUnit
629 .....	7-4-5-قراردادهای نام گذاری JUnit برای Maven
629 .....	7-4-6-مجموعه تست های JUnit (JUnit test suites)
630 .....	7-4-7-اجرای تست از طریق خط فرمان/ command line
631 .....	7-5-5-ساختارهای پایه ای فریم ورک JUnit
631 .....	7-5-1-JUnit های Annotation
632 .....	7-5-2-دستورات Assert
633 .....	7-5-3-ترتیب اجرای تست
634 .....	7-5-4-غیرفعال سازی تست ها
634 .....	7-6-پشتیبانی محیط برنامه نویسی از Eclipse
634 .....	7-6-1-ایجاد تست های JUnit
635 .....	7-6-2-اجرای تست های JUnit
638 .....	7-6-3-استخراج و بازیابی تست های ناموفق و stacktrace ها
638 .....	7-6-4-امکان static import در فریم ورک JUnit
639 .....	7-6-5-برنامه‌ی راهنمای Wizard برای ساختن مجموعه تست (test suite)
640 .....	7-6-6-تست exception و خطاهای exception
641 .....	7-6-7-تست نویسی برای افزونه ها (JUnit Plug-in Test)
641 .....	7-7-نصب JUnit
641 .....	7-7-1-استفاده از JUnit با سیستم کامپایل Gradle
641 .....	7-7-2-استفاده از JUnit با سیستم کامپایل Maven
642 .....	7-7-3-استفاده از JUnit درون ساخته‌ی محیط کاری Eclipse
642 .....	7-7-4-دانلود کتابخانه‌ی JUnit
642 .....	7-8-تنظیم محیط برنامه نویسی Eclipse برای استفاده از امکان static import کتابخانه‌ی JUnits

646 .....	7-ساخت یک کلاس Java
647 .....	7-ساخت و طراحی یک تست JUnit
651 .....	7-اجرای تست در محیط برنامه نویسی Eclipse
652 .....	7-تنظیمات و امکانات پیشرفته JUnit
653 .....	7-9-1-اجرای تست های دارای پارامتر مشخص (Parameterized test)
655 .....	7-9-2-دستور @Rule (annotation)
656 .....	7-9-3-طراحی و تنظیم rule های اختصاصی برای JUnit
658 .....	7-9-4-Category ها و دسته بندی تست ها
658 .....	7-10-ایجاد آبجکت های ساختگی یا شبیه سازی رفتار آبجکت/Mocking
659 .....	بخش دوم :
660 .....	7-10-1-مقدمه ای بر تست اپلیکیشن های اندرویدی
661 .....	7-10-2-بخش هایی که در تست نرم افزاری، تمرکز بر روی آن قرار می گیرد
663 .....	7-10-3-پیش شرط های تست گیری (testing preconditions)
663 .....	7-10-4-ATSL و تست اپلیکیشن با ایزار 4 JUnit
665 .....	7-11-ساختار پروژه اندرویدی و ایجاد پوششی تست
665 .....	7-11-1-سازماندهی پروژه اندرویدی برای تست
665 .....	7-11-2-برطرف کردن خطای "error duplicate files in path"
666 .....	7-12-اجرای Unit test بر روی JVM
666 .....	7-12-1-اجرای Unit test بر روی نرم افزار در بستر Android runtime
666 .....	7-12-2- محل جایگذاری unit test ها در پروژه اندرویدی
667 .....	7-12-3-کتابخانه های الزامی در فایل dependency/Gradle build
667 .....	7-12-4-اجرای unit test ها از طریق سیستم کامپایل Gradle
667 .....	7-12-5-اجرای unit test ها از محیط کاری Android Studio
668 .....	7-12-6- محل قارگیری نتایج و گزارش های مربوط به تست (test reports)
669 .....	7-12-7-فعال سازی مقدایر بازگشتی پیش فرض متدهای ساختگی (mocked methods) در فایل android.jar
670 .....	7-12-8-افزودن dependency / کتابخانه JUnit
671 .....	7-12-9-اجرای unit test بر روی پروژه
671 .....	7-13-1-طراحی instrumentation test برای اجرای تست در بستر دستگاه حقیقی اندروید
671 .....	7-13-1-Instrumentation test
673 .....	7-13-2-سیستم اندروید چگونه تست ها را اجرا می کند
673 .....	7-13-3-شبیه سازی رفتار آبجکت ها در اندروید (ایجاد آبجکت های ساختگی جهت تست)
674 .....	7-13-4- محل قارگیری تست های instrumentation

674 .....	7-معرفی dependency ها و testInstrumentationRunner داخل فایل build.gradle	13-5
674 .....	7-استفاده از @RunWith(AndroidJUnit4.class)	13-6
675 .....	7-اجرای unit test از طریق سیستم کامپایل Gradle	13-7
675 .....	7-اجرای unit test از داخل محیط برنامه نویسی Android Studio	13-8
675 .....	7-محل جایگذاری گزارش های تست	13-9
675 .....	7-نحوه ی جایگزین کردن بخش های اپلیکیشن با instrumentation test	13-10
676 .....	7-ایجاد کلاسی جهت تست گیری	13-11
677 .....	7-ساخت unit test جدید	13-12
678 .....	7-اطلاعات بیشتر در خصوص اجرای تست بر روی اپلیکیشن های اندرویدی	13-14
678 .....	7-کلاس های assert	14-1
678 .....	7-Test group ها (گروه بندی تست ها)	14-2
679 .....	7-فیلتر کردن تست	14-3
680 .....	7-دستور @FlakyTest	14-4
681 .....	7-ساخت کلاس و پروژه ی تست	14-5
682 .....	7-استفاده از ابزار Monkey جهت ایجاد و ارسال شبه event به دستگاه	15-15
682 .....	7-توضیحی درباره ی ابزار monkey	15-1
683 .....	7-نحوه ی استفاده از ابزار Monkey	15-2
683 .....	7-تست آبجکت Application	16
685 .....	7-معرفی unit test برای تست آبجکت application	16-1
685 .....	7-طرایی instrumentation test برای آبجکت application	16-2
686 .....	7-تست سایر کامپوننت های نرم افزاری اندروید	17
686 .....	7-تست سرویس	17-1
687 .....	7-تست کامپوننت نرم افزاری Content provider	17-2
687 .....	7-تست Loader	17-3
688 .....	7-تест های مترقبه یا پشتیبان Annotation	18
688 .....	7-Annotation های مربوط به Nullness	18-1
688 .....	7-Annotation های مربوطه به thread	18-2
689 .....	7-تلهیه ی گزارش از مقدار کد تست شده از کل پروژه (Code coverage report)	18-3
691 .....	7-ساخت پوشه ی تست در محیط کاری Android Studio	19
697 .....	بخش سوم :	
697 .....	7-تست بخش های مجازی نرم افزار با استفاده از آبجکت های ساختگی (mock objects)	20
697 .....	7-هدف از نوشتن unit test و چالش هایی که در تست با آن ها مواجه می شوید	20-1

698.....	7- طبقه بندی کلاس های تست گیری مختلف
699.....	7- نحوه ی ساخت mock object
700.....	7- استفاده از Mockito برای شبیه سازی و ایجاد آبجکت ساختگی
701.....	7- افزودن Mockito در قالب dependency به پروژه
701.....	7- استفاده از سیستم کامپایل Gradle
701.....	7- استفاده از سیستم کامپایل Maven
701.....	7- استفاده از محیط کاری Eclipse
702.....	7- افزونه نویسی برای Eclipse (چارچوب توسعه و نصب کتابخانه ها و کامپوننت های نرم افزاری)
703.....	7- استفاده از توابع کتابخانه ای Mockito API / Mockito
703.....	7- دستورات Static import
704.....	7- اعلان و تنظیم آبجکت های ساختگی / mock objects با فریم ورک Mockito
705.....	7- تنظیم آبجکت های ساختگی / mock ها
707.....	7- متدهای verify() (بررسی صحت فراخوانی یک متدهای پارامترهای مورد نظر)
708.....	7- قرار دادن آبجکت های جاوا در Spy (رصد و شنود آبجکت های جاوا و توابع فراخوانده شده بر روی آن با spy)
708.....	7- پیاده سازی الگوی dependency injection در فریم ورک Mockito با استفاده از دستور @InjectMocks
709.....	7- دسترسی به آرگومان های ارسالی به توابع (Capturing arguments)
710.....	7- محدودیت های فریم ورک Mockito
710.....	7- استفاده از فریم ورک Mockito در اندروید
711.....	7- اضافه کردن dependency مورد نیاز Mockito به فایل app/build.gradle
712.....	7- ساخت یک unit test جدید
713.....	7- ایجاد یک نمونه Twitter API
713.....	7- شبیه سازی از نمونه ITweet
714.....	7- بررسی صحت فراخوانی متدهای
714.....	7- مرحله ای اعتبار سنجی تست
714.....	7- استفاده از PowerMock به همراه Mockito
714.....	7- استفاده از PowerMock برای شبیه سازی متدهای static
715.....	7- استفاده از یک Powermock wrapper بجای
716.....	7- بخش چهارم :
716.....	7- هدف از Hamcrest matcher framework
718.....	7- استفاده از Hamcrest matchers

718.....	اعلان های مربوطه برای dependency	Gradle-7-27-1
718.....	اعلان های لازم برای سیستم dependency	Maven-7-27-2
719.....	اضافه کردن Hamcrest به طور مستقیم به classpath	Eclipse-7-27-3
720.....	استفاده کاربردی از Hamcrest	Hamcrest-7-28
720.....	استفاده از Hamcrest های matcherjlist ها	7-28-1
721.....	مرور کلی بر Hamcrest های matcher	Hamcrest-7-28-2
722.....	ساخت Hamcrest matcher اختصاصی خود	7-29
724.....	بخش پنجم :	
724.....	مقدمه ای بر AssertJ	AssertJ-7-30
725.....	استفاده ای کاربردی از AssertJ	7-31
725.....		Gradle-7-32
725.....		Maven-7-31-2
725.....	تنظیمات مرتبط با محیط کاری Eclipse	7-31-3
725.....	تنظیمات مرتبط با محیط کاری IntelliJ	7-31-4
726.....	بخش ششم :	
727.....	فریم ورک تست گیری Espresso و تست لایه‌ی UI اپلیکیشن	Espresso-7-32
729.....	نصب و استفاده از Espresso	7-33
729.....	نصب	7-33-1
729.....	تنظیم فایل Gradle build برای استفاده از توابع Espresso	7-33-2
730.....	تنظیمات دستگاه (Device settings)	7-33-3
733.....	تنظیم و ویرایش فایل app build.gradle	7-33-4
733.....	تست نویسی برای پروژه بر اساس فریم ورک Espresso	7-33-5
734.....	ساخت و تنظیم intent start intent (اجرا و راه اندازی دوم) activity	7-33-6
735.....	ضبط تعامل با UI اپلیکیشن و جهت اجرای تست Espresso (Espresso UI recorder)	7-33-7
737.....	تنظیم activity مورد تست	7-33-8
739.....	اجرای تست های Espresso	7-34
739.....	اجرای تست در محیط کاری Android Studio	7-34-1
740.....	استفاده از سیستم کامپایل Gradle	7-34-2
741.....	بررسی صحت نمایش پیغام toast	7-34-3
742.....	شبیه سازی آجکت های Intent با توابع کتابخانه‌ی Espresso	7-35
743.....	نوشتن تست برای بررسی عملکرد intent	7-36
743.....	ایجاد پروژه‌ی آزمایشی (project under test)	7-36-1

745 .....	7-36-2-بررسی صحت تست (اعتبارسنجی)
746 .....	7-36-3-طراحی تست های functional برای activity ها
747 .....	7-37-آزمایش عملکرد کد ناهزمان با استفاده از فریم ورک تست گیری Espresso
752 .....	بخش هفتم :
752 .....	7-38-تست تعامل بین چندین کامپوننت نرم افزاری اپلیکیشن با استفاده از فریم ورک UI Automator
753 .....	7-39-تست تعامل بین کامپوننت های نرم افزاری اپلیکیشن به روشن blackbox
753 .....	7-39-1-استفاده از UI Automator جهت تست تعامل بین کامپوننت های اپلیکیشن (تست کل اپلیکیشن)
754 .....	7-39-2-دسترسی به اطلاعات مربوط به view با ابزار uiautomatorviewer
757 .....	7-39-3- محل قرارگیری تست ها
757 .....	7-39-4-نحوه ی نوشتن تست
757 .....	7-39-5-ساخت پروژه و تنظیم فایل Gradle build
760 .....	بخش هشتم :
761 .....	7-41-نصب Robotium
762 .....	7-42-نمونه ای از پیاده سازی تست های Robotium
763 .....	7-43-توابع کتابخانه ای / Robotium API
766 .....	7-43-1-ساخت پروژه ی آزمایشی و اضافه کردن کتابخانه ی Robotium به آن
766 .....	7-43-2-ایجاد پروژه ی آزمایشی و افزودن کتابخانه ی Robotium
768 .....	بخش نهم :
769 .....	7-44-مقدمه ای بر مفهوم dependency injection <> استفاده از dependency injection در محیط و بستر اجرای Java
769 .....	7-44-1-Dependency injection چیست؟
771 .....	7-44-2-استفاده از annotation ها جهت توصیف و شرح نیازمندی ها / dependency های کلاس
772 .....	7-44-3-بر اساس JSR330، آبجکت ها به کدام بخش از کلاس تزریق یا پاس داده می شوند؟
773 .....	7-44-4-ترتیبی که DI بر اساس آن در کلاس اجرا و پیاده سازی می شود.
774 .....	7-45-فریم ورک های جاوا و dependency injection
774 .....	7-46-Dependency injection با فریم ورک Dagger2
774 .....	7-46-1-Dagger2 چیست؟
775 .....	7-46-2-اعلان dependency provider ها (تعریف ارائه دهنگان آبجکت های موردنیاز و نیازمندی های کلاس)....
776 .....	7-46-3-تعریف آبجکت های مورد نیاز یا dependency consumer ها (Object consumer)
776 .....	7-46-4-برقراری ارتباط بین Consumer ها (کلاس های درخواست کننده یا مصرف کننده) و provider های (ارائه دهنگان) آبجکت های مورد نیاز
777 .....	7-46-5-Scope annotation

777	7-برخورد ویژه‌ی Dagger با فیلدها (field injection)	7-46-6
778	7-استفاده از 2 Maven در محیط کاری Eclipse همراه با سیستم کامپایل Dagger	7-47-7
780	7-ساخت پروژه‌ی آزمایشی و استفاده از فریم‌ورک 2 Dagger	7-46-7
780	7-معرفی یا تنظیم و ویرایش فایل Maven build	7-46-8
781	7-معرفی و استفاده از کلاس‌هایی که به آبجکت‌هایی از بیرون نیاز دارند (درخواست dependency دارند) ... (@Component دستور	7-46-9
782	7-استفاده از کد تولید شده توسط Dagger	7-46-11
783	7-47 دو دستور دیگر در اندروید	7-47
783	7-پیاده‌سازی الگو توسعه‌ی DI در پروژه‌های اندرویدی	7-47-1
783	7-استفاده‌ی کاربردی از 2 Dagger در اندروید	7-47-2
786	7-افزودن dependency‌ها و کتابخانه‌های مورد نیاز سیستم کامپایل Gradle	7-47-3
787	7-ترسیم یا اعلان نمودار dependency‌ها و نمایش وابستگی آبجکت‌های مورد نیاز به یکدیگر	7-47-4
788	7-ساخت آبجکت Application پروژه‌ی (Wiring up the application)	7-47-5
789	7- تست اپلیکیشن و کسب اطمینان از عملکرد صحیح اپلیکیشن	7-47-6
791	7-جایگزین کردن @Module با کلاس‌های اختصاصی خود در تست	7-48
792	بخش دهم :	
792	7-مرور کلی	7-49
792	StrictMode	7-50
794	7-50-1 مقدمه	
795	7-50-2 استفاده از ابزار TraceView در محیط کاری Android Studio	7-50-7
797	7-50-3 راه اندازی و فراخوانی ابزار TraceView از طریق خط دستور (command line)	
801	7-50-4 رصد و سنجش کارایی اپلیکیشن (Trace)	
801	7-50-5 برطرف کردن کاستی‌های مربوط به کارایی اپلیکیشن	
802	7-51 Hierarchy Viewer (ابزار نمایش درختی view‌ها)	
804	7-52 بینه‌سازی Layout	
807	7-53 تهیه‌ی روگرفت از حافظه (Memory Dumps)	
807	7-54 ابزار Systrace	
811	7-55 شبیه‌سازی چگالی یا تراکم پیکسلی (pixel density) از طریق خط دستور	
811	7-56 قالب‌های آماده یا الگوهای پیاده‌سازی پروژه‌ی اندروید (Android template)	
811	7-57 گزارش گیری و ثبت اطلاعات مربوط به کارایی موتور گرافیکی (ابزار Profile GPU rendering)	
812	7-58 تحلیل و بررسی وضعیت Overdraw (ترسیم یک آیتم بر روی آیتم دیگر)	
813	7-59 ابزار رایگان و متن باز کارا برای تحلیل و بینه‌سازی کارایی اپلیکیشن	

814 .....	7-استفاده از Leak Canary برای یافت و برطرف نمودن هدر رفت حافظه (memory leak)
814 .....	7-استفاده از AndroidDevMetrics جهت مشاهده ای اطلاعات مربوط به کارایی اپلیکیشن
817 .....	بخش اول :
817 .....	8-استفاده از Gradle برای کامپایل پروژه های اندرویدی
817 .....	8-1-فایل های build اپلیکیشن های اندروید
818 .....	8-1-2-پروسه ای که برای تبدیل به اپلیکیشن اندروید طی می کند
821 .....	8-1-3-resource shrinking
821 .....	8-1-4-تعريف dependency ها و اعلان ورژن کتابخانه های لازم خارج از بدنه ای بستار (closure)
822 .....	8-2-build و کامپایل ورژن های flavor مختلف از اپلیکیشن اندرویدی خود
822 .....	8-2-1-انواع build (build flavor)
823 .....	8-2-2-تعريف انواع flavor در فایل Gradle build
824 .....	8-2-3-ارائه ای منابع مختلف برای flavor ها و ورژن های مختلف اپلیکیشن
824 .....	8-2-4-تعريف source set های متفاوت برای flavor های مختلف از اپلیکیشن
828 .....	8-2-5-اعتبارسنجی و تست پروژه
829 .....	8-2-6-کامپایل و build پروژه از طریق خط دستور Gradle
829 .....	8-2-7-تست ورژن یا flavor های مختلف از یک اپلیکیشن (gradle flavor های یک اپلیکیشن)
829 .....	8-2-8-پیاده سازی نسخه های مختلف از کلاس MainActivity در mord نظر از اپلیکیشن
830 .....	8-3- تنظیم اختصاصی فایل build Gradle
830 .....	8-3-1-ویرایش اسم فایل apk خروجی
830 .....	8-3-2-تعريف keystore مجزا برای debug build
831 .....	8-4-انتقال / migrate کردن یک پروژه ای خروجی گرفته شده از محیط Eclipse به Gradle
831 .....	8-4-1-وارد کردن (import) یک پروژه ای خروجی گرفته شده از Eclipse در محیط کاری Android Studio
831 .....	8-4-2-اضافه کردن فایل Gradle به پروژه ای اندرویدی که از محیط Eclipse خروجی گرفته شده است
832 .....	بخش دوم :
833 .....	8-5-کامپایل و اجرای پروژه های اندرویدی با Jenkins
833 .....	8-5-1-پیش نیاز های کامپایل پروژه (build job) با Jenkins
833 .....	8-5-2-نصب مجموعه ابزار ساخت و توسعه ای نرم افزار اندروید (Android SDK)
834 .....	8-5-3-نصب افرونه ها یا Jenkins plug-in های
834 .....	8-5-4-ایجاد build job برای اپلیکیشن های اندرویدی
837 .....	8-5-5-اجرای تست بر روی دستگاه
838 .....	8-5-6-دیگر افزونه های مفید برای build پروژه های اندرویدی

840.....	بخش سوم :	
استفاده از ابزار Android Debug Bridge (پل برقراری ارتباط و کنترل دستگاه یا شبیه ساز محیط اندرویدی و خط از دایی کد)		
840.....		
آموزش حاضر شرح می دهد چگونه با استفاده از ابزار ADB به دستگاه واقعی یا شبیه ساز محیط اندروید (AVD) متصل شده و آن را مدیریت کنید.		
840.....		
6-ابزار دسترسی، مدیریت و اشکال زدایی پروژه ای اندرویدی / ADB		
840.....		
841.....	8-استفاده از adb	
841.....	8-7-اجرای یک activity از خط دستور (command line)	
841.....	8-8-حذف اپلیکیشن از دستگاه به وسیله ی دستورات adb	
842.....	8-9-اتصال به دستگاه با استفاده از Telnet	
842.....	8-10-دریافت اطلاعات سیستمی با ابزار خط دستور dumpsys	
843.....	8-10-1-دستور adb dumpsys	
843.....	8-10-2-مروری بر میزان مصرف حافظه با دستور dumpsys	
843.....	8-10-3-واکنشی اطلاعات درباره ی عملیات و تسلک های زمان بندی شده	
845.....	بخش اول :	
846.....	توابع کتابخانه ای مکان یابی اندروید / Android Location API	
846.....	846.....	بدست آوردن اطلاعات مربوط به مکان جغرافیایی دستگاه
846.....	846.....	نصب
846.....	استفاده از LocationManager	
847.....	بدست آوردن مختصات جغرافیایی به صورت دو طرفه (geocoding)	
847.....	امنیت .....	
847.....	درخواست از کاربر برای فعال سازی حسگر GPS (امکان سخت افزاری GPS یا مکان یاب)	
848.....	استفاده از امکان GPS و تنظیم موقعیت جاری	
848.....	فعال سازی امکان GPS در محیط شبیه ساز	
849.....	تنظیم موقعیت جغرافیایی	
850.....	آموزش: استفاده از توابع کتابخانه ای Location API اندروید	
854.....	بخش دوم :	
854.....	Google Maps	
854.....	MapsView	
855.....	MapFragment	
856.....	کلاس marker (نشانه) بر روی نقشه .	

856	تنظیم اختصاصی GoogleMap
857	نرم افزار شبیه ساز محیط اندروید (emulator) و Google Maps
857	نصب Google Play service
858	ساخت و دریافت کلید شناسگر (key)
858	ایجاد SHA-1 برای امضای دیجیتالی (Signature key)
859	ثبت نام در Google APIs Console
859	ساخت کلید و امضای دیجیتالی برای اپلیکیشن
867	بخش اول :
867	امکان ساخت افزاری Camera
873	بخش دوم :
873	پیاده سازی کلاس SensorManager و دسترسی به حسگرهای دستگاه اندروید
874	گوش فرادهنده به تغییرات حسگر (Sensor listener)
875	آموزش استفاده ای کاربردی از حسگر Accelerometer
877	آموزش کاربردی: ساخت یک قطب نما
880	بخش سوم :
880	رخدادهای مربوط به لمس نمایشگر (Android touch)
880	واکنش نشان دادن به event های مربوط به touch
882	وضعیت لمس یک نقطه از نمایشگر (single touch)
883	قابلیت لمس چندین نقطه از صفحه (Multi touch)
885	کلاس GestureDetectors
890	تمرین: پیاده سازی قابلیت پشتیبانی از لمس چندین نقطه از نمایشگر (Multitouch)
894	آموزش: پیاده سازی قابلیت بزرگ نمایی / کوچک نمایی و استفاده از multitouch
896	بخش چهارم :
897	کار با Gesture ها و پیاده سازی کلاس GestureOverlayView در اندروید
903	بخش اول :
903	استفاده از Google های Support library
903	شرح مفهوم Support Library
904	نصب Support library
904	Support library هایی که توسط گوگل ارائه می شوند
906	حذف support library از پروژه
906	چرا باید support library را حذف کرد؟
906	تبديل پروژه به یک پروژه ای استاندارد اندروید

909 .....	بخش دوم :
909 .....	پروژه های کتابخانه ای اندروید و کتابخانه های جاوا .....
909 .....	استفاده از کتابخانه های اندروید و جاوا .....
910 .....	نحوه استفاده از فایل JAR در اپلیکیشن اندرویدی .....
910 .....	حدودیت هایی در استفاده از کتابخانه های java .....
910 .....	ماژول های اختصاصی کتابخانه ای اندروید (library modules) .....
910 .....	ماژول های کتابخانه ای (library module) .....
911 .....	اولویت در انتخاب منابع (conflicting resources) .....
911 .....	ایجاد ماژول های کتابخانه ای اختصاصی در اندروید (Android library module) .....
912 .....	ایجاد یک کتابخانه ای جدید (library module) .....
914 .....	ایجاد کلاس model (data model) .....
915 .....	ایجاد کلاس های مورد نیاز (instances) .....
915 .....	اضافه کردن کتابخانه به عنوان dependency به پروژه .....
918 .....	استفاده از پروژه کتابخانه جهت بروز رسانی داده های details fragment ها .....
919 .....	تست اپلیکیشن و بررسی عملکرد صحیح آن .....
919 .....	بخش سوم :
920 .....	پروژه های کتابخانه ای کارامد و پرکاربرد اندروید .....
920 .....	لیست کتابخانه های پرکاربرد اندروید .....
921 .....	Code repository و انباره نمونه کدها و پروژه ها .....
921 .....	بخش چهارم :
922 .....	نصب کتابخانه .....
923 .....	چه زمانی باید از Otto استفاده کرد؟ .....
923 .....	چگونگی تنظیم Otto .....
923 .....	نحوه register یا گوش دادن به event ها و unregister آن ها .....
924 .....	نحوه ارسال event ها .....
924 .....	چگونه کامپوننت های نرم افزاری جدید می توانند آخرین event را دریافت کنند؟ .....
930 .....	بخش اول .....
930 .....	مدیریت صدا در محیط اندروید (Android Sound) .....
930 .....	پخش صدا .....
931 .....	کلاس MediaRecorder .....
931 .....	افزودن Media library جدید به .....
932 .....	فرمت های مورد پشتیبانی .....

932 .....	آموزش کاربردی: پخش صدا به وسیله‌ی کلاس SoundPool
934 .....	آموزش کاربردی: ضبط صدا (media) با استفاده از کلاس MediaRecorder
936 .....	بخش دوم
936 .....	دریافت کد برنامه (Android source code)
936 .....	نصب ابزار مورد نیاز
937 .....	کپی کردن کد اصلی اپلیکیشن (source code cloning)
937 .....	ابزار ساخت و توسعه‌ی اپلیکیشن‌های اندرویدی (ADT)
938 .....	کمک به توسعه‌ی پروژه ADT
938 .....	نحوه‌ی ساخت ابزار
938 .....	بخش سوم
938 .....	برنامه نویسی و توسعه اپلیکیشن برای Android
938 .....	سیستم عامل اندروید
939 .....	سیستم داخلی اندروید (internals)
942 .....	بخش چهارم
942 .....	Cloud to device messaging
942 .....	مقایسه‌ی poll و push
942 .....	سرвис ارسال اطلاعات از server به اپلیکیشن‌های اندرویدی / سرویس Cloud
944 .....	ابزار لازم برای تست کاربردی C2DM
944 .....	مجوزهای لازم
945 .....	اعلان intent receiver (جهت دریافت intent های مربوطه)
945 .....	ثبت و معرفی app server (سرвис دهنده‌ی اپلیکیشن)
947 .....	درباره شناسه‌ی ثبت (registration ID) اپلیکیشن تحت موبایل
950 .....	ثبت و اعلان یک Receiver ویژه‌ی پیغام‌های ارسالی از سرورهای C2DM
951 .....	ارسال پیغام
953 .....	ثبت نام و درخواست برای استفاده از سرویس C2DM
953 .....	ایجاد پروژه و فایل layout
954 .....	ایجاد receiver ها و activity های مورد نیاز
960 .....	ثبت اپلیکیشن برای دریافت پیغام (Registration)
964 .....	بخش پنجم
964 .....	توابع کتابخانه‌ای مربوط به تقویم / آموزش Calendar API

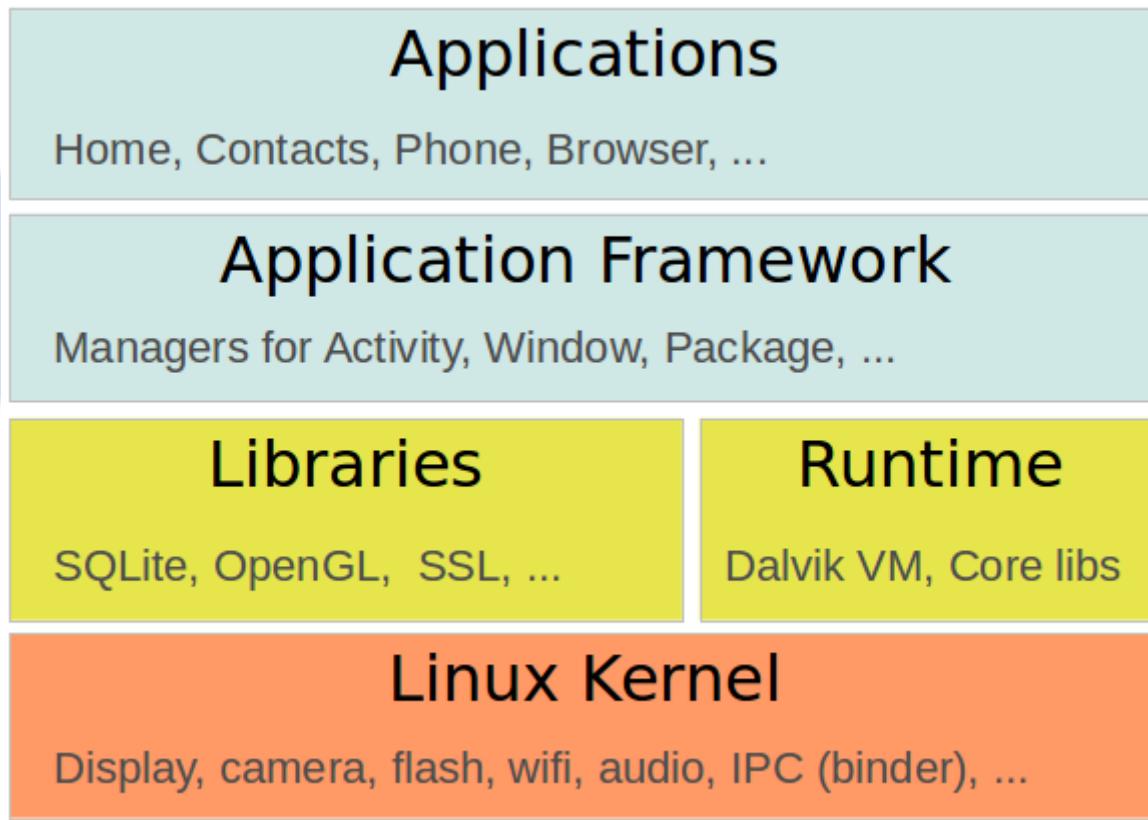


# 1-1-مقدمه ای بر برنامه سازی تحت موبایل برای سیستم عامل اندروید

## 1-1-1-سیستم عامل اندروید

یک سیستم عامل تحت موبایل و مبتنی بر هسته Linux است. پروژه‌ی توسعه و ارتقا اندروید AOSP یا پروژه‌ی کد باز اندروید نام دارد که شرکت Google آن را رهبری می‌کند.

سیستم عامل اندروید از چهار لایه تشکیل شده است، اما یک توسعه دهنده‌ی اندروید معمولاً با دو لایه‌ی بالابی (Application Framework و Application) آن سروکار دارد.



## معماری اندروید به شرح زیر می باشد:

- لایه ی Application – پروژه ی کد باز اندروید دربردارنده ی چندین اپلیکیشن پیش فرض همچون مرورگر وب، اپلیکیشن استفاده از دوربین (Camera)، اپلیکیشن پخش موسیقی Music، اپلیکیشن برقراری تماس Phone و غیره ... می باشد. در واقع تمامی برنامه هایی که کاربران از آن ها استفاده می کنند در این لایه نصب شده و در درسترس کاربر قرار می گیرند.
- لایه ی Application framework – یک API است که امکان برقراری تعامل سطح بالا از اپلیکیشن ها با سیستم اندروید را فراهم می آورد. این لایه عمدتی خدمات سطح بالا و مورد نیاز اپلیکیشن ها را در قالب کلاس های جاوا فراهم آورده و به برنامه نویس اجازه ی بهره گیری از این امکانات در توسعه ی اپلیکیشن را می دهد.
- لایه ی Libraries – کتابخانه هایی که انجام کارهای معمول نظیر نمایش و ارائه ی خروجی گرافیکی (rendering)، ذخیره ی داده ها، قابلیت ویگردی را به صورت آماده فراهم می آورند. این لایه همچنین Runtime اندروید و کتابخانه های اصلی که امکان اجرای اپلیکیشن ها را فراهم می آورد، شامل می شود.
- لایه ی هسته ی Linux – این لایه امکان ارتباط با سخت افزار را مهیا می سازد. تاکنون ویرایش های متعددی از اندروید منتشر شده که در جدول زیر آن ها را مشاهده می کنید:

Code name	Version	API level
Nougat	N	24
Marshmallow	6.0	23
Lollipop	5.1	22
Lollipop	5.0	21
KitKat	4.4 - 4.4.4	19

Jelly Bean	4.1.x - 4.3.x	16 - 18
Ice Cream Sandwich	4.0.1 - 4.0.4	14 -15
Honeycomb	3.2.x	13
Honeycomb	3.0 - 3.1	11 - 12
Gingerbread	2.3 - 2.3.7	9-10
Froyo	2.2.x	8
Eclair	2.1	7
Eclair	2.0 - 2.0.1	5 -6
Donut	1.6	4
Cupcake	1.5	3
(no code name)	1.1	2
(no code name)	1.0	1

## 2-1-2-نحوه‌ی برنامه‌سازی برای سیستم عامل تحت موبایل اندروید

زبان برنامه نویسی که برای ساخت و توسعه‌ی اپلیکیشن اندروید بکار گرفته می‌شود، زبان شی گرا و همه منظوره‌ی Java است. به عبارت دیگر، برنامه نویس اندروید تمامی فایل‌های تنظیمات

و منطق اپلیکیشن که برای یک برنامه ی کاربردی اندروید ضروری است را با زبان چند منظوره ی Java می نویسد.

Android development tooling (مجموعه ابزار ساخت و توسعه ی اپلیکیشن های اندروید) فایل هایی که توسط محیط کاری و برنامه نویس تولید شده را به اپلیکیشن اندروید تبدیل می کند. حال اگر برنامه نویس فرایند deployment را آغاز کند، کل اپلیکیشن اندروید به زبان ماشین ترجمه، کلاس های اپلیکیشن پوشه بندی و package شده (تحت یک namespace یا پوشه ی واحد سازمان دهی شده)، سپس برای اجرا تنظیم و آماده می شود (و در نهایت به اجرا در می آید).

فرایندی است که طی آن اپلیکیشن برای بهره برداری کاملاً آماده می شود.)

SDK (مجموعه ابزار ساخت و توسعه ی اپلیکیشن اندروید) و Gradle (سیستم ترجمه و کامپایل کدها به زبان ماشین) ابزار لازم جهت ایجاد، کامپایل و package کردن (سازمان دهی یا دسته بندی کلاس ها تحت یک namespace واحد) اپلیکیشن های اندروید را بر عهده دارد. تیم توسعه دهندگان اندروید افزونه ی Gradle را جهت هدایت اپلیکیشن های اندروید به مرحله ی کامپایل فراهم می آورد که می توانید آن را از اینترنت دانلود کرده و بر روی محیط کاری Android Studio نصب نمایید.

SDK یا مجموعه ابزار ساخت و توسعه ی اپلیکیشن های اندروید، ابزار ADB (پل ارتباطی به دستگاه اندروید و اشکال زدایی اپلیکیشن) را نیز شامل می شود. ADB یک پل ارتباطی به دستگاه های واقعی و مجازی اندروید جهت مدیریت یا اشکال زدایی اپلیکیشن می باشد.

### ADT-1-1-3 / مجموعه ابزار ساخت و توسعه ی اپلیکیشن اندروید و محیط Android Studio

شرکت Google جهت توسعه و تست اپلیکیشن های اندروید، IDE یا محیط برنامه نویسی Android Studio را ارائه کرده و توصیه می کند. این محیط کاری خود از محیط برنامه نویسی IntelliJ برگرفته شده است.

در ADT ویرایشگرهای اختصاصی ویژه‌ی فایل‌های اندروید تعییه شده. بیشتر فایل‌های تنظیمات اندروید مبتنی بر XML هستند. ویرایشگرهایی که از آن‌ها نام برده شد به شما امکان می‌دهند بین نسخه‌ی XML (حالت ویرایش) فایل و رابط کاربری ساخت یافته که امکان وارد کردن داده را مهیا می‌کند، راه گزینی (سویچ) نمایید.

## 1-1-پروسه‌ی تبدیل و ترجمه از کد برنامه به اپلیکیشن اندروید

**توجه:** مطالبی که در بخش حاضر مطالعه می‌کنید مربوط به فرایند کامپایل بوده و برای توسعه‌ی اپلیکیشن ضروری نمی‌باشد. در صورت تمایل می‌توانید از آن رد شوید.

کامپایلر Java فایل‌های حاوی کد و دستورات (source file) جاوا را به فایل‌های کلاس (class file) تبدیل می‌کند. در واقع SDK اندروید در خود ابزاری به نام dx دارد که فایل‌های Java را به یک فایل اجرایی با پسوند .dex (Dalvik Executable) تبدیل می‌کند. تمامی فایل‌های کلاس اپلیکیشن داخل این فایل اجرایی قرار داده می‌شوند. در طول پروسه‌ی تبدیل، اطلاعات تکراری و غیر ضروری موجود در فایل‌های کلاس حذف شده و به طور بهینه سازمان دهی می‌شوند. به طور مثال، اگر یک متغیر String هم‌زمان در چندین فایل کلاس وجود داشته باشد، فایل .dex تنها یک اشاره گر (reference) به این String را در خود نگه می‌دارد.

با توجه به توضیح بالا، فایل‌های .dex بسیار کم حجم‌تر از فایل‌های کلاس متناظر هستند.

فایل .dex و محتویات یک پروژه‌ی اندروید (resource) همچون تصاویر و فایل‌های XML همگی به صورت پکیج داخل فایل .apk (Android package) قرار داده می‌شوند. این عملیات را برنامه‌ای به نام aapt انجام می‌دهد.

فایل apk. خروجی، تمامی داده های لازم برای اجرای اپلیکیشن را شامل می شود. حال کافی است این فایل را به وسیله ای ابزار adb (پل ارتباطی مدیریت دستگاه اندروید و اشکال زدایی اپلیکیشن) بر روی دستگاه اندروید نصب (deploy) کرد.

از ویرایش 5.0 به بعد اندروید، ART به عنوان سیستم مدیریت اجرای برنامه برای تمامی اپلیکیشن های اندروید مورد استفاده قرار می گیرد. Ahead Of Time compilation ART از (ترجمه ی کل کد برنامه به زبان ماشین به صورت یکجا در ابتدای اجرای اپلیکیشن) بهره می گیرد. حین نصب یک اپلیکیشن بر روی دستگاه اندروید، کد آن اپلیکیشن به زبان ماشین ترجمه می شود. این امر سبب می شود کد پس از کامپایل با افزایش 30 درصدی حجم مواجه شود، اما در زمان راه اندازی برنامه سرعت اجرا را بالا می برد.

از آنجایی که کد اپلیکیشن تنها به هنگام اجرا و راه اندازی اولیه اپلیکیشن به زبان ماشین ترجمه می شود، مصرف باتری به مراتب کاهش می یابد.

ابزار dex2oat فایل dex. تولید شده را به فرمت ELF تبدیل می کند. این فایل کد dex، کد ترجمه شده به زبان ماشین که مستقیماً توسط پردازنده اجرا می شود (native code) و meta-data (اطلاعاتی پیرامون اپلیکیشن) را شامل می شود. فایل با نگه داشتن کد dex. امکان استفاده از ابزار جاری را همچنان فراهم می آورد.

عملیات مدیریت حافظه (garbage collection) در ART بهینه سازی شده، از این جهت مدت زمانی که اپلیکیشن ممکن است به طور موقت کند شده یا متوقف شود، کاهش می یابد.

## 1-1-5- فروشگاه مجازی Google Play

شرکت گوگل سرویسی به نام Google Play را ارائه می دهد که این سرویس یک فروشگاه مجازی است و برنامه نویسان می توانند اپلیکیشن خود را در آن برای استفاده ی کاربران به صورت رایگان و پولی عرضه کنند. کاربران، اپلیکیشن Google Play را نصب کرده و به واسطه ی آن برنامه های کاربردی دلخواه خود را از سرویس Google play دانلود و نصب می کنند.

اپلیکیشن Google Play یک سرویس ویژه کاربران ارائه می دهد که آن ها را از آپدیت و نسخه‌ی جدید یک نرم افزار آگاه می سازد. برای مثال زمانی که توسعه دهنده یک نسخه‌ی جدید از اپلیکیشن خود را در فروشگاه بارگذاری می کند، این سرویس بلافاصله کاربران جاری خود را از آن آپدیت مطلع ساخته و این آپدیت را در اختیار آن ها قرار می دهد.

Google play همچنین کتابخانه‌ها و سرویس‌هایی را ویژه‌ی برنامه سازان اندروید ارائه می دهد. به عنوان مثال می توان به سرویسی اشاره کرد که امکان استفاده و نمایش Google Maps را مهیا می سازد. ارائه‌ی این سرویس‌ها از طریق Google Play این مزیت را دارد که ویرایش‌های قدیمی اندروید نیز می توانند از آن ها استفاده کنند. Google می تواند آن ها را بدون اینکه نیازی به بروز آوری کلی سیستم عامل اندروید باشد، بروز رسانی کند.

## 2-1-نصب محیط برنامه نویسی

### 2-1-1-سیستم مورد نیاز برای نصب Android Studio

می توان بر روی یک سیستم سخت افزاری متوسط به راحتی برای سیستم عامل اندروید برنامه نویسی کرد. هر چند برای شرایط مطلوب، بهتر است محیط سخت افزاری مدرن با پردازنده‌ی 2.6 گیگاهرتز و رم 8 گیگابایت استفاده نمایید. هارد SSD نیز می تواند به شدت در اجرا و راه اندازی سریع برنامه شبیه ساز اندروید موثر باشد.

### 2-1-2-ابزار لازم برای نصب و استفاده از سیستم عامل Linux

SDK اندروید 32 بیتی است، به همین جهت بر روی یک سیستم لینوکس 64 بیتی، می بایست پکیج ia32-libs را نصب نمایید. در نسخه‌ی Ubuntu می توانید این کار را با اجرای دستور زیر انجام دهید:

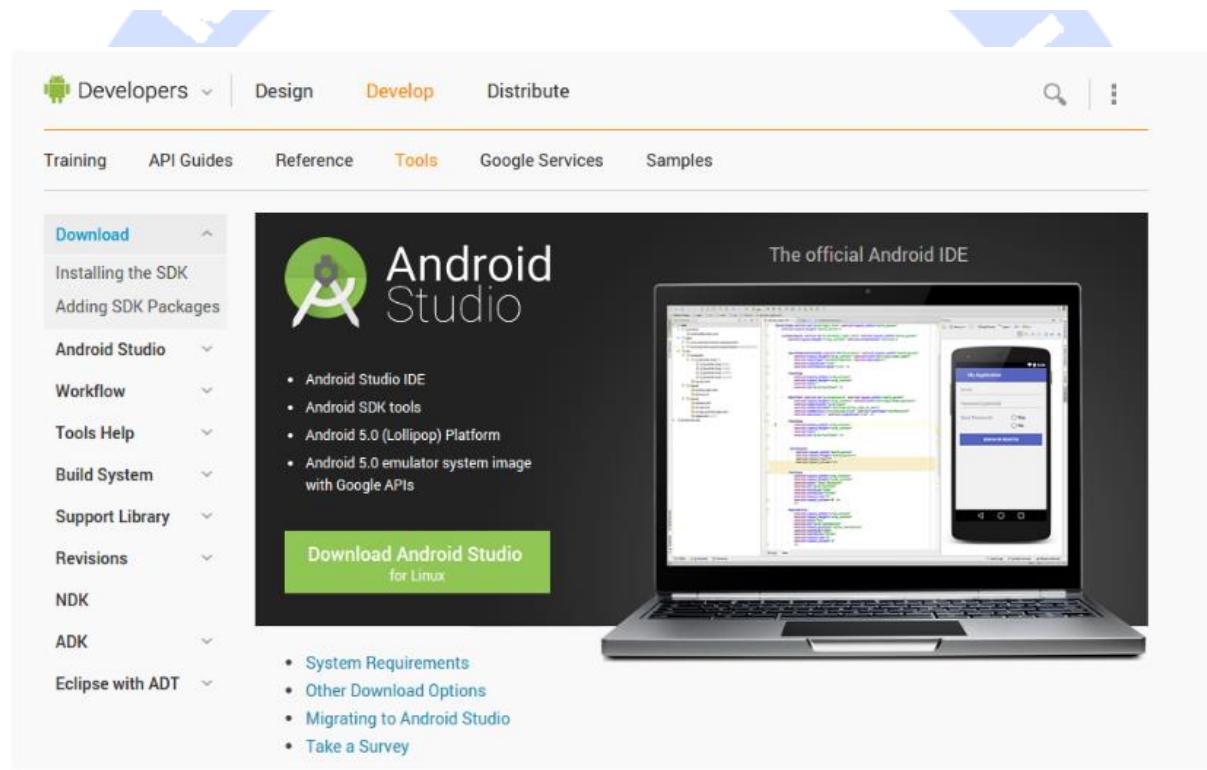
```
apt-get install ia32-libs
```

چنانچه از نسخه‌ی دیگر Linux استفاده می کنید، توصیه می شود توضیحات و مستندات ارائه شده همراه با آن نسخه را در خصوص دستور مربوط مطالعه نمایید.

## 3-2-1-دانلود Android Studio از اینترنت

می توانید [Android Studio](http://developer.android.com/sdk/index.html) را از آدرس <http://developer.android.com/sdk/index.html> دانلود نمایید.

دانلود در دو نسخه عرضه می شود: 1. یک نسخه فقط SDK Tools را شامل می شود. 2. دیگری کل Android Studio Packages را دربر می گیرد. شما می بایست Package را دانلود نمایید.

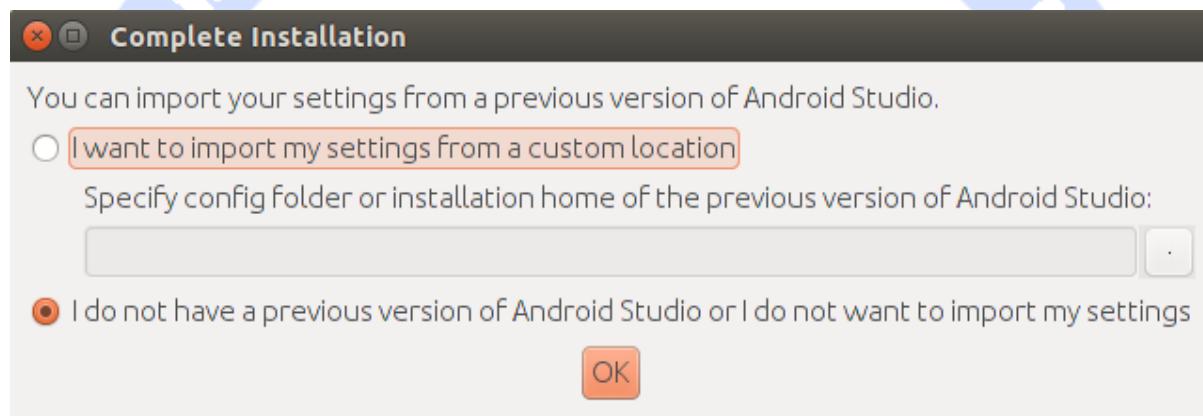


## 4-2-1-نصب محیط برنامه نویسی Android Studio

پروسه نصب بر روی سیستم عامل Windows آسان است، کافی است فایل اجرایی .exe را که از اینترنت دانلود کردید، اجرا نمایید. در سیستم عامل Max OSX کافی است Android Studio را با اشاره گر موس کشیده و در پوشه‌ی Applications جایگذاری نمایید.

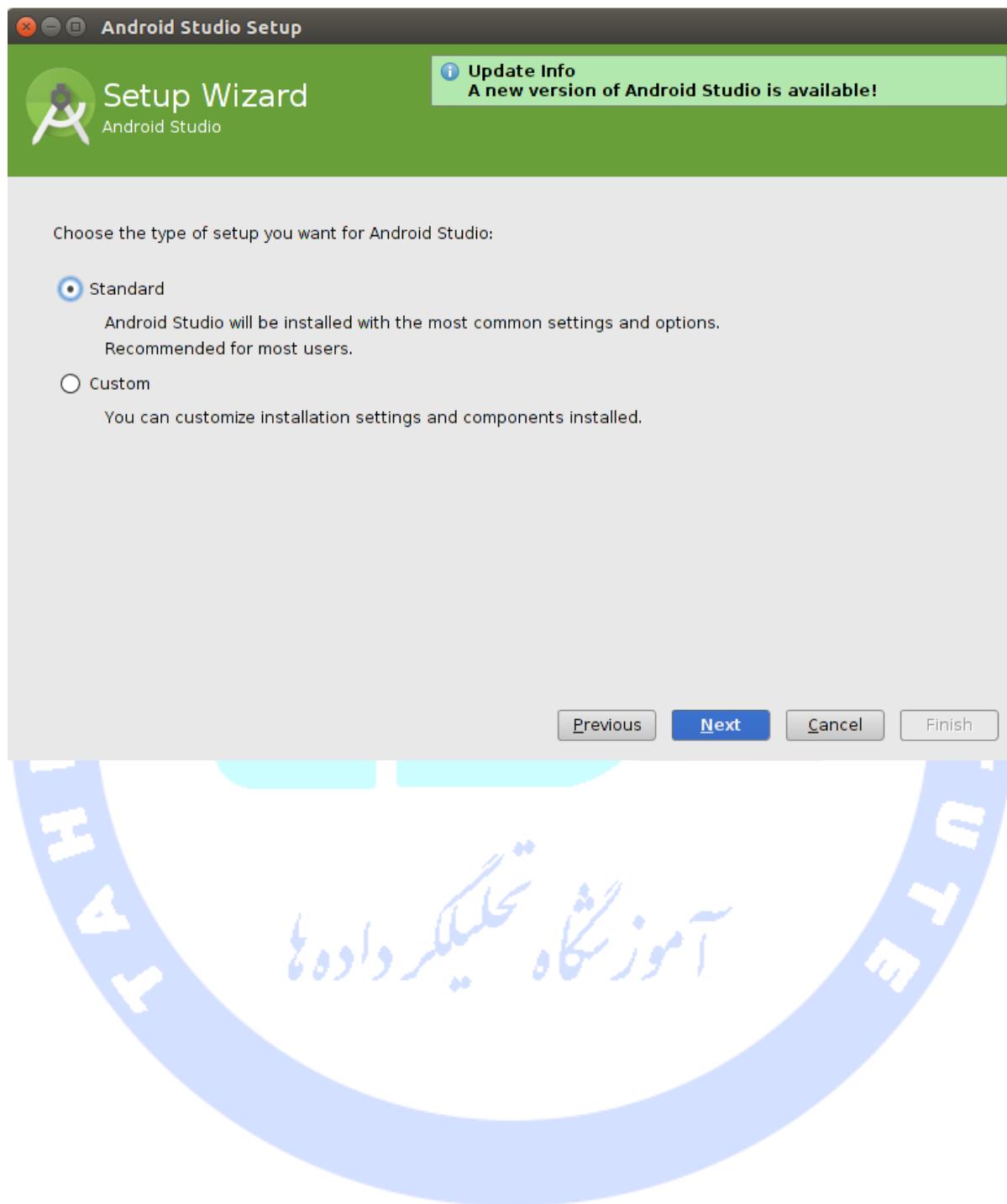
در سیستم عامل Linux می بایست فایل ZIP دانلود شده را از حالت فشرده خارج ساخته و در مکان مناسب جایگذاری نمایید. به منظور راه اندازی پروسه نصب Android Studio، ابتدا به آدرس android-studio/bin داخل یک پنجره terminal فرمان (terminal) پیمایش نموده و سپس دستور studio.sh را اجرا نمایید.

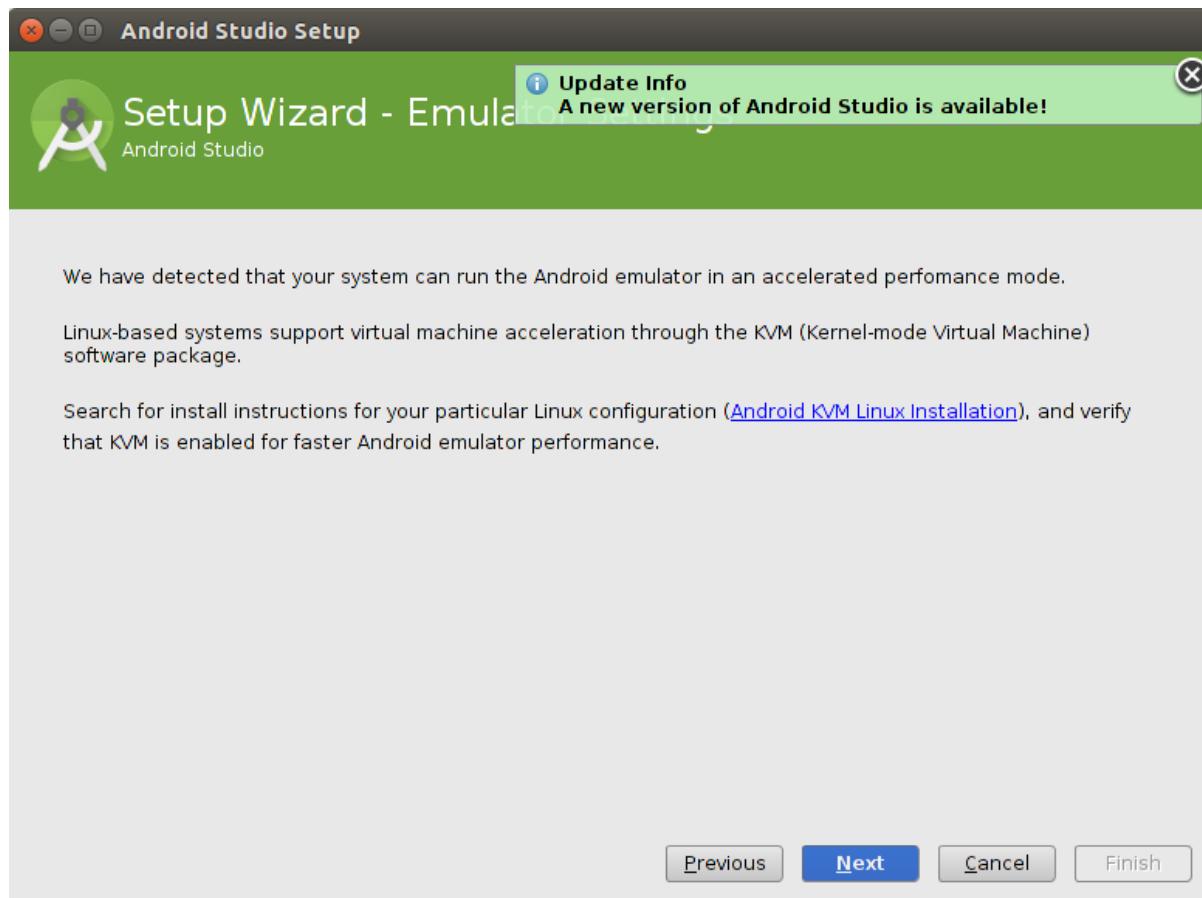
زمانی که پروسه نصب Android Studio را راه اندازی می کنید، این امکان در اختیار شما قرار می گیرد که تنظیمات مورد نیاز نسخه نصبی موجود را (برای مثال setting ویرایش قدیمی تر که قبلاب روی سیستم خود نصب کردید) به محیط جدید وارد نمایید.



باقی مراحل نصب را به صورت ویژاردی دنبال نمایید.

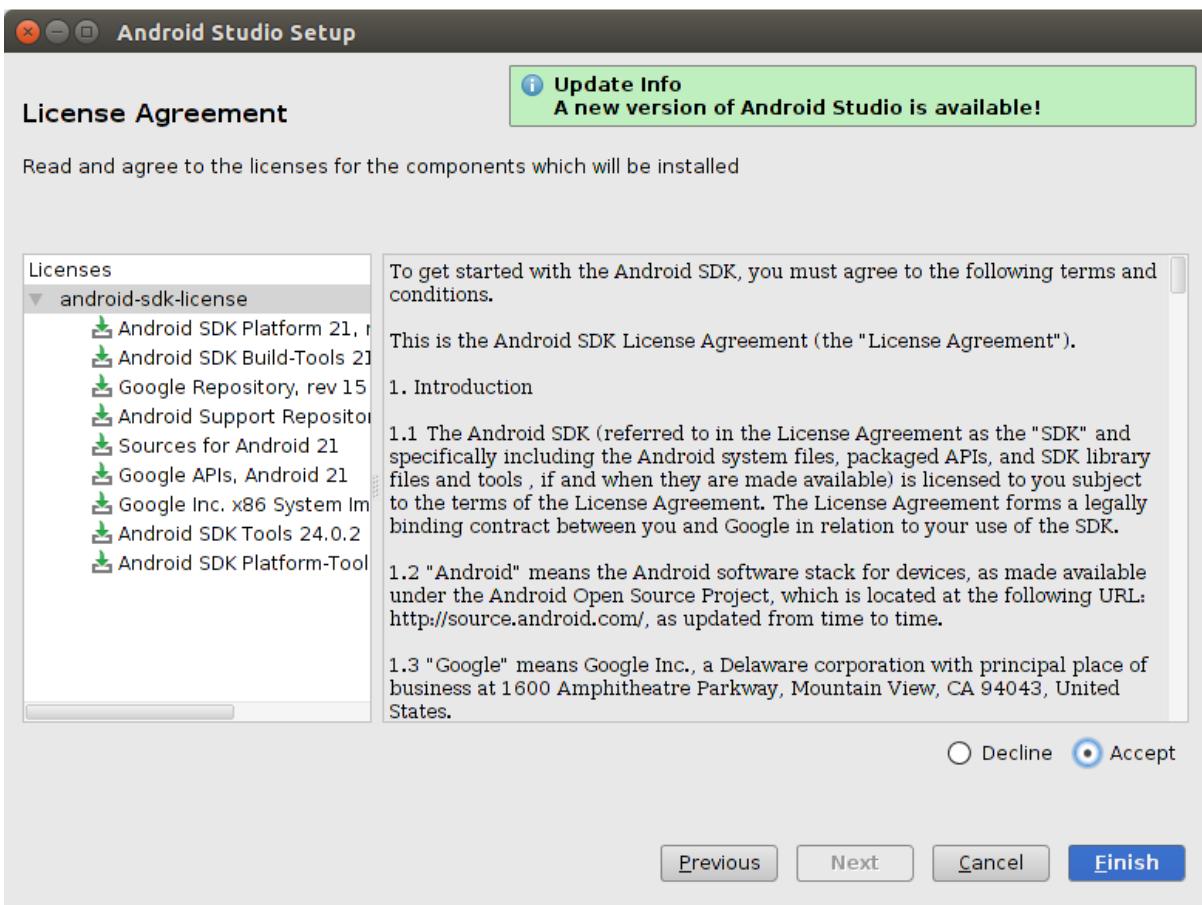






در آخرین صفحهٔ نصب، بر روی دکمهٔ **Finish** کلیک نمایید.





## تمرین: شروع به کار با Android Studio

در این تمرین می‌توانید یک پروژه‌ی اندروید تعریف کرده، سپس دستگاه مجازی (VM) را بر روی آن راه اندازی نمایید.

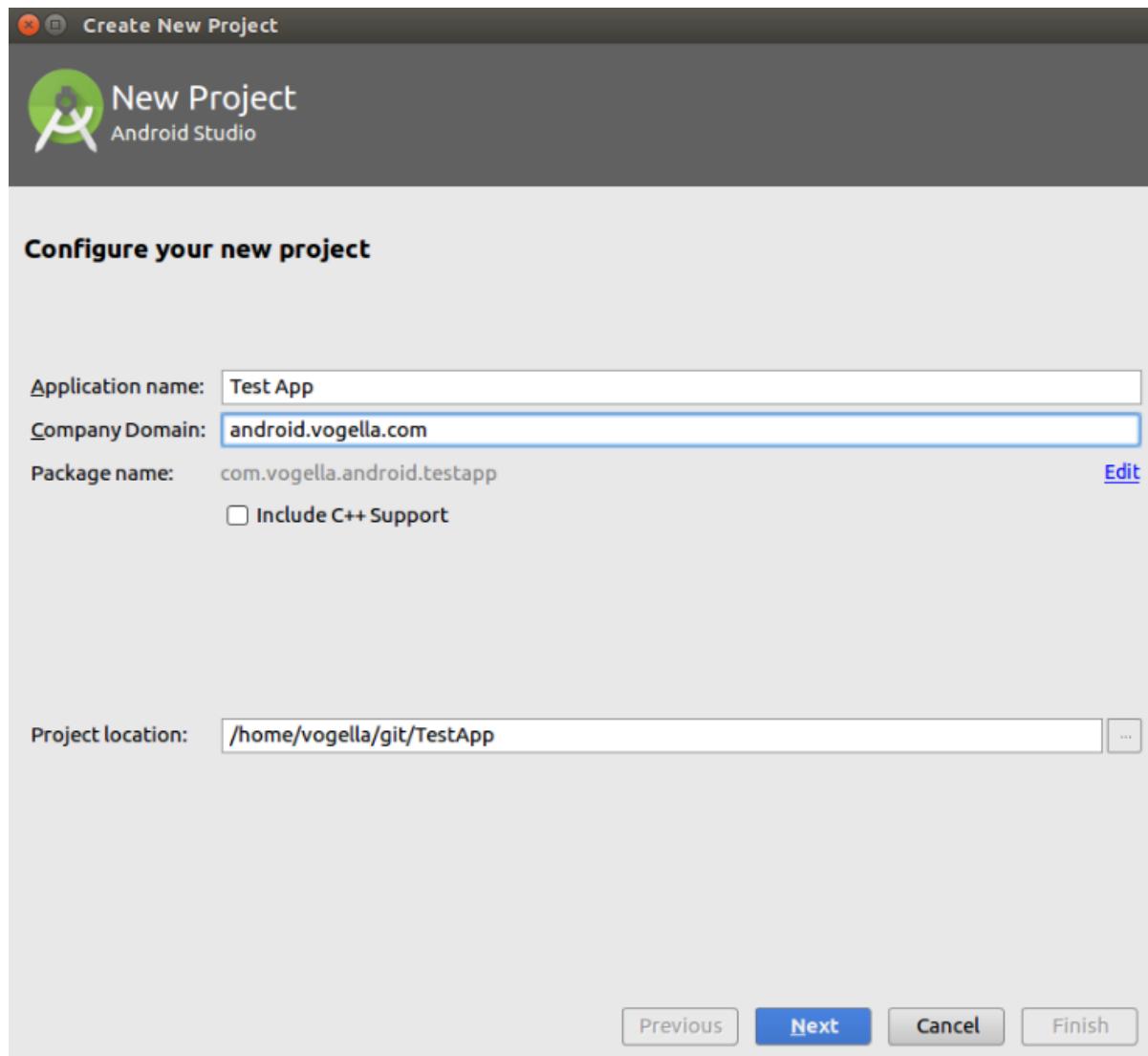
جهت ساخت یک اپلیکیشن ساده‌ی اندروید، می‌توانید بر روی لینک Start a new Android Studio project کلیک نمایید و یا اگر قبلاً یک پروژه‌ی اندروید ایجاد کردید، این مسیر را طی نمایید: [File > New Project](#).

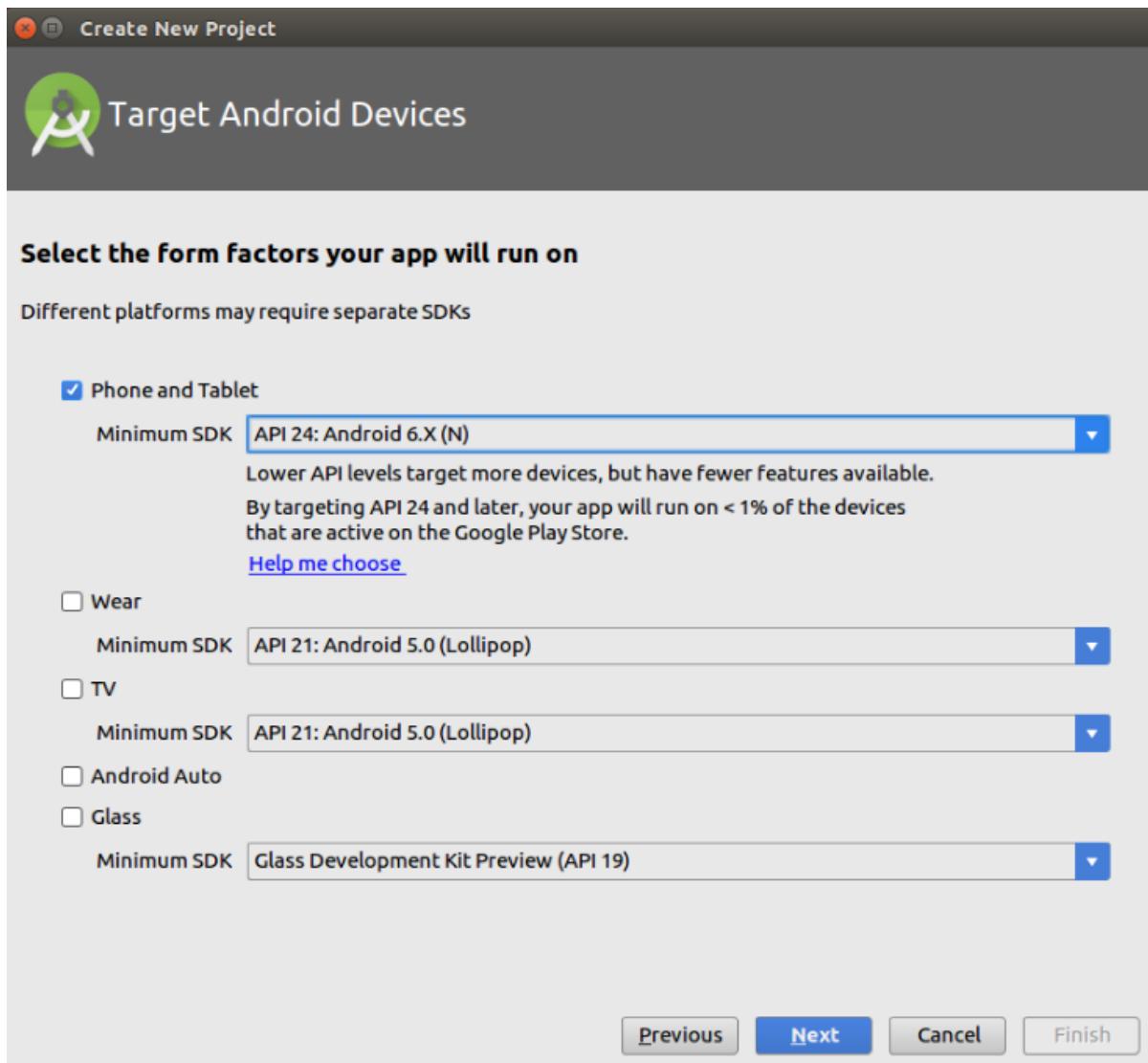


Property	Value
Application name	Test App

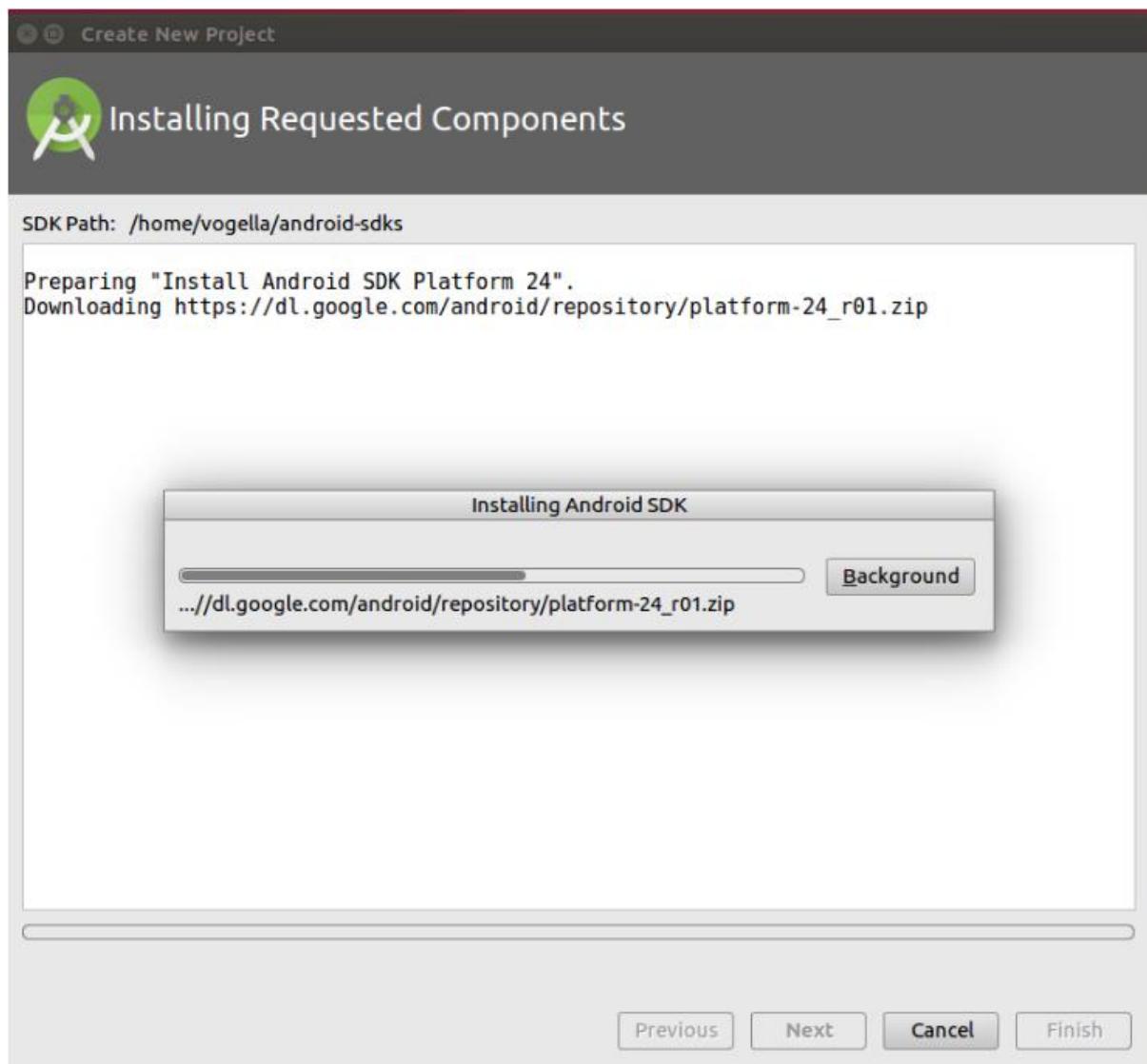
Property	Value
Company Domain	android.vogella.com
Package Name	com.vogella.android.testapp
API (Minimum, Target, Compile with)	Latest
Template	Empty Activity



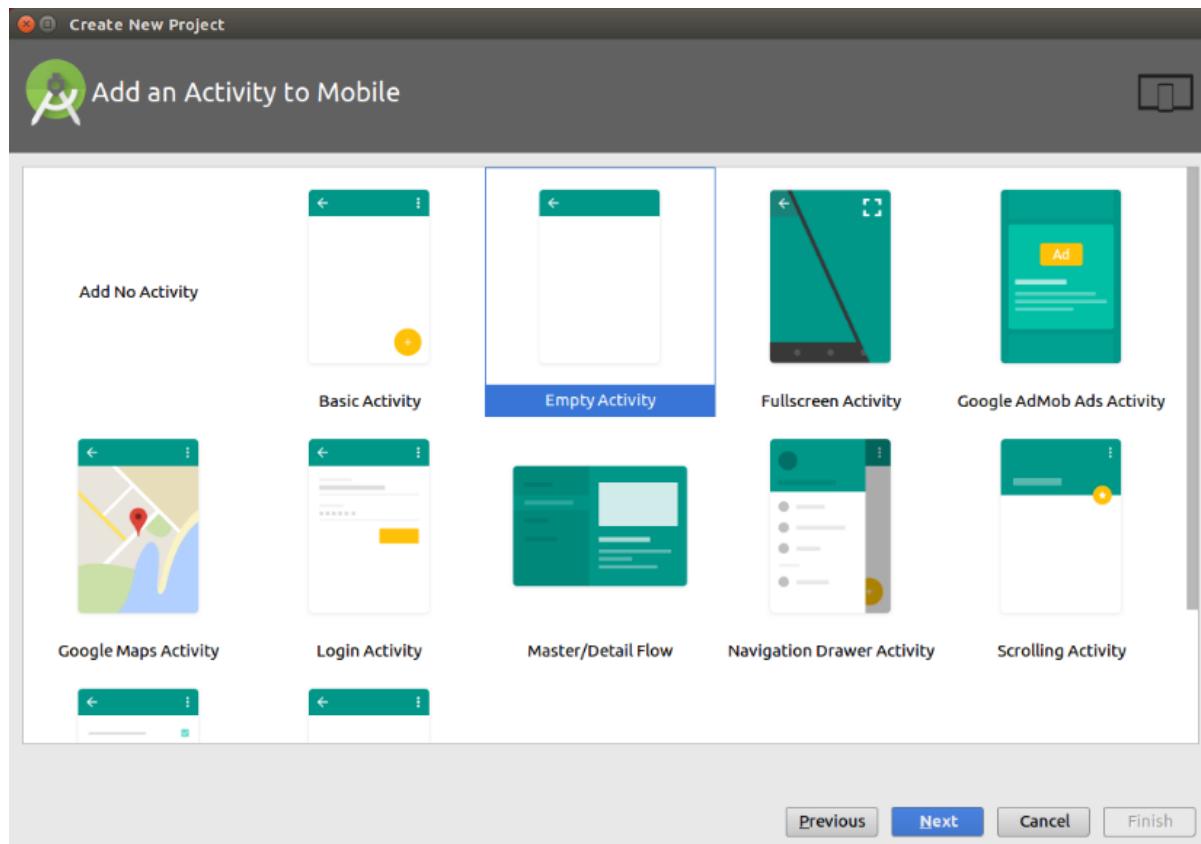




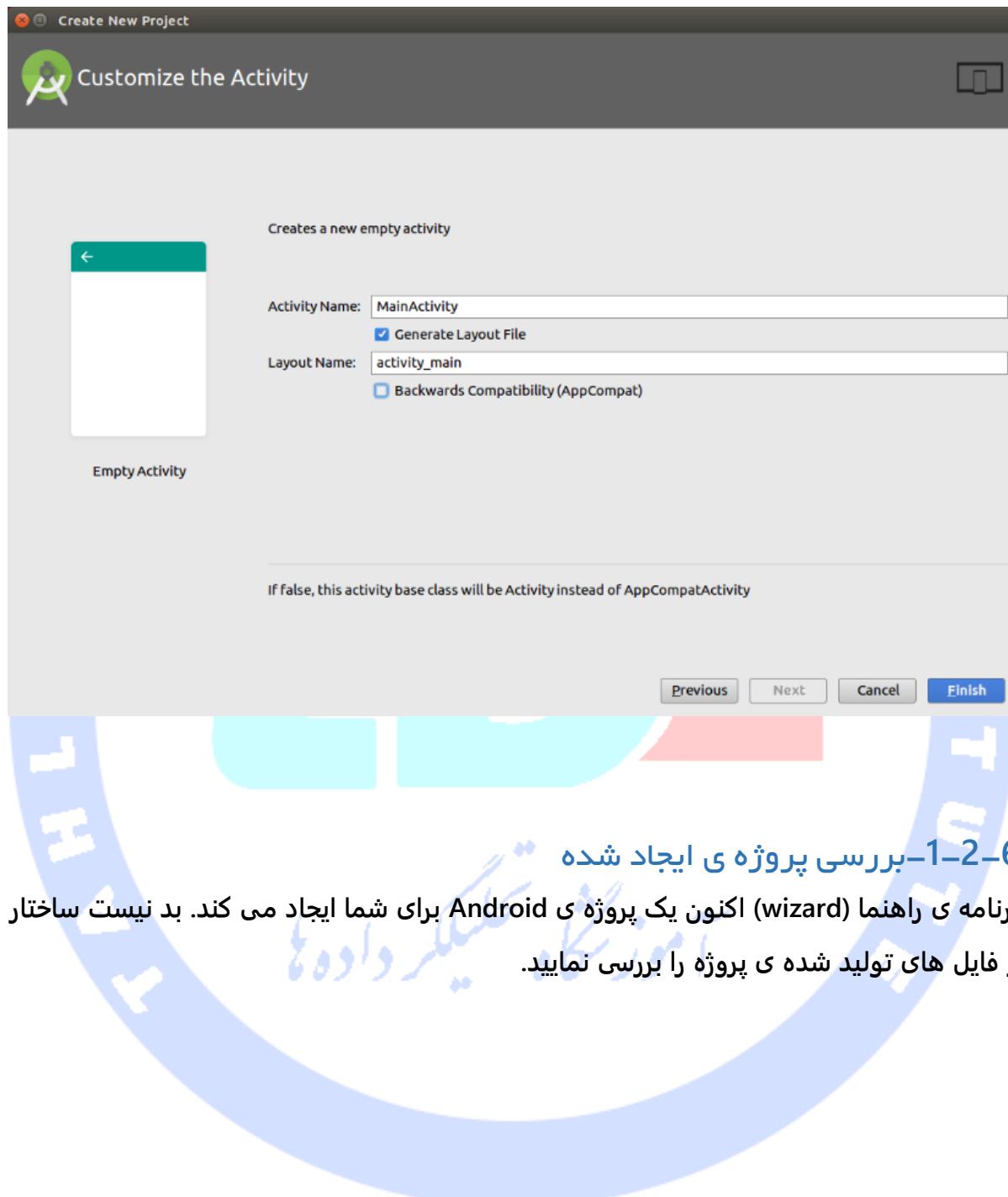
اگر قبل از SDK را دانلود نکرده باشید، Android Studio به صورت خودکار SDK مورد نیاز را از اینترنت بارگیری می کند.



در پنجره‌ی Empty Activity، قالب آماده‌ی Add an Activity to Mobile را انتخاب نمایید.

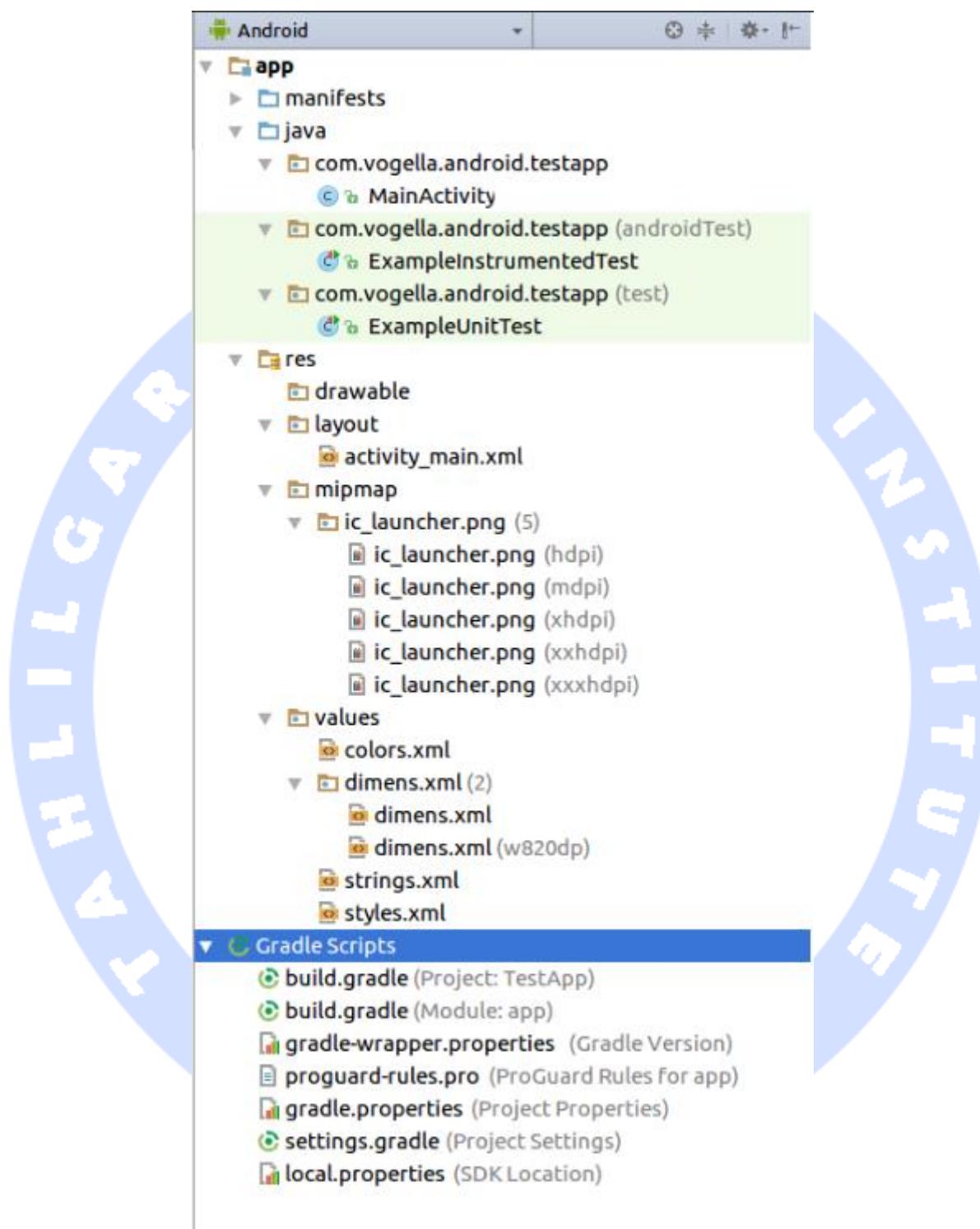


در آخرین صفحه ی تنظیمات پروژه ی جدید، گزینه ی Backwards Compatibility (قابلیت پشتیبانی از ویرایش های قبلی) را از حالت انتخاب خارج نمایید.



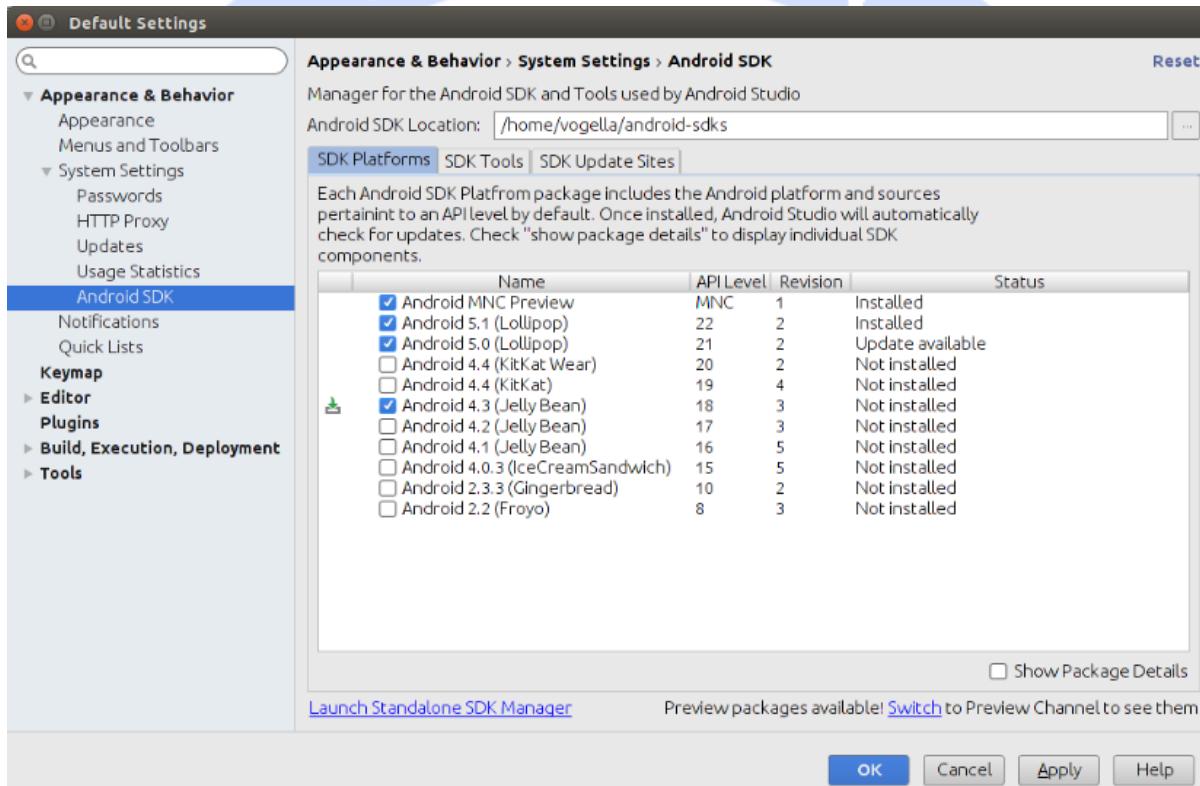
## ۱-۲-۶-بررسی پروژه‌ی ایجاد شده

برنامه‌ی راهنمای (wizard) اکنون یک پروژه‌ی Android برای شما ایجاد می‌کند. بد نیست ساختار و فایل‌های تولید شده‌ی پروژه را بررسی نمایید.



## 1-2-7-نصب نسخه‌ی مورد انتخاب سیستم عامل اندروید

در SDK Manager، نسخه‌ی دلخواه اندروید برای نصب را از نمای درختی انتخاب نموده و سپس دکمه‌ی Install را کلیک نمایید. همان طور که در تصویر زیر مشاهده می‌کنید، API 18 از کتابخانه‌های اندروید انتخاب شده و نصب می‌شود.



با کلیک بر روی دکمه‌ی OK پروسه‌ی نصب را آغاز نمایید.

در تب SDK Platforms، می‌توانید ورژن API یا کتابخانه‌های اندروید را برای نصب انتخاب نمایید.

در تب SDK Tools نیز می‌توانید ابزار توسعه و ساخت اپلیکیشن را نصب نمایید.

## ۱-۲-۸-نصب support library های جدید (کتابخانه‌ی پشتیبانی از API های جدید) اپلیکیشن در نسخه‌های قدیمی‌تر آن)

این امکان را برای شما فراهم می‌آورد تا از قابلیت‌ها و امکانات ویرایش‌های بالاتر اندروید در نسخه‌های پایین‌تر آن استفاده نمایید. علاوه بر آن، این کتابخانه قابلیت‌هایی که به صورت جدا از Android عرضه می‌شود (unbundled)، همچون ابزارک (widget) (widget) RecyclerView جهت نمایش بهینه‌ی لیست، را در اختیار شما قرار می‌دهد.

در حال حاضر ویرایش‌های متعددی از این کتابخانه وجود دارد: v4، v7 و v13. این کتابخانه‌ها هر یک برای ورژن‌های مربوطه قابل استفاده می‌باشند. به عنوان مثال، support library v7 برای دستگاه‌های اندروید که از کتابخانه‌های ورژن 7 (API 7) اندروید استفاده می‌کنند، قابل استفاده می‌باشد. لازم به ذکر است که نسخه‌های بالاتر support library برای عملکرد صحیح خود نیاز به ورژن‌های پایین‌تر دارند. به طور مثال، support library v7 برای عملکرد صحیح خود به کتابخانه v4 احتیاج دارد.

## ۱-۳-۱-تست اپلیکیشن‌های اندروید بر روی محیط مجازی (ADV) یا دستگاه واقعی

### ۱-۳-۱-محیط شبیه ساز اندروید (emulator) و دستگاه مجازی اندروید (ADV)

مجموعه ابزار اندروید (Android Tooling) در خود یک نرم افزار شبیه ساز (Android device) دارد. این شبیه ساز ADV را اجرا می‌کند. emulator ADV می‌تواند محیط و سخت افزار یک گوشی واقعی اندروید را شبیه سازی کند.

AVD ها این امکان را برای شما فراهم می‌کنند تا اپلیکیشن‌های اندروید خود را بر روی ورژن‌های مختلف اندروید با تنظیمات سخت افزاری مختلف تست کنید. حتی اگر هم یک دستگاه واقعی اندروید برای تست اپلیکیشن‌های خود دارید، لازم است نحوه کار با ADV ها را بیاموزید چرا که

دستگاه های مجازی به شما اجازه می دهند برنامه‌ی کاربردی خود را بر روی ورژن‌های مختلف اندروید با تنظیمات سخت افزاری خاص اجرا و تست نمایید.

تنظیمات دلخواه دستگاه مجازی را می بایست در طول پروسه‌ی ایجاد AVD مشخص نمایید. این تنظیمات انتخاب کیفیت تصویر (resolution)، ورژن کتابخانه‌های اندروید (API version) و تراکم پیکسلی مد نظر را شامل می شود.

می توانید چندین AVD با تنظیمات مختلف تعریف نمایید و آن‌ها را همزمان اجرا کنید. این کار به شما اجازه می دهد به طور همزمان اپلیکیشن خود را بر روی چندین دستگاه با تنظیمات مختلف امتحان کنید.

**نکته:** اگر AVD را حین راه اندازی اولیه متوقف نمایید، دستگاه مجازی خراب می شود. راه اندازی اولیه در دستگاه‌های قدیمی کمی زمان می برد. در دستگاه‌های جدید فرایند راه اندازی اولیه معمولاً ۱ تا ۳ دقیقه طول می کشد.

پس از اجرای کامل AVD، می توانید GUI را به وسیله‌ی موس کنترل نمایید. شبیه ساز همچنین با ارائه‌ی منوی در سمت راست محیط، به شما امکان می دهد تا به دکمه‌های گوشی دسترسی داشته باشید. سعی کنید در طول پروسه‌ی توسعه‌ی اپلیکیشن، AVD را متوقف نکنید. چنانچه در اپلیکیشن خود تغییراتی را اعمال نموده و هم اکنون می خواهید نسخه‌ی جدیدی از آن را تست نمایید، بهتر است اپلیکیشن خود مجدداً بر روی AVD مستقر (deploy) کنید.

## Debug certificate-1-3-2 (شناختن‌نامه‌ی تاریخ تولید و ثبت اپلیکیشن) و تاریخ ابطال آن (expiry date)

اپلیکیشن‌های اندروید می بایست قبل از نصب بر روی دستگاه، امضا شده و کلید منحصر بفردی به آن تخصیص داده شود. در واقع در طول پروسه‌ی توسعه‌ی برنامه، محیط کاری Eclipse اپلیکیشن شما را به صورت خودکار با یک certificate یا امضا دیجیتالی خاص به نام debug key نشانه‌گذاری می کند تا امکان کپی برداری از آن وجود نداشته باشد.

حال این امضای دیجیتالی که به برنامه ی کاربردی تخصیص داده می شود، حدوداً تا 365 روز اعتبار دارد. زمانی که این تاریخ به پایان می رسد، اپلیکیشن یک خطای زمان کامپایل (build error) صادر کرده و به شما اعلان می کند که certificate یا امضای دیجیتالی تخصیص داده شده به اپلیکیشن منقضی شده و دیگر معتبر نیست.

به منظور رفع این مشکل، کافی است فایل debug.keystore را حذف نمایید. محل ذخیره ی فایلی های اپلیکیشن به صورت پیش فرض در سیستم عامل X OS و Linux در آدرس C:\Documents و در سیستم عامل ویندوز XP در آدرس C:\Documents and Settings\[username]\.android\7 و در Vista و در آدرس C:\Users\[username]]\.\.android\

دفعه ی بعد که اپلیکیشن را کامپایل (build) می کنید، ابزار کامپایل پروژه یک keystore key store = جهت تایید debug key جدید برای اپلیکیشن تولید کرده و به آن اختصاص می دهد (id توسعه دهنگان مورد استفاده قرار می گیرد).

### Android AVD در مقایسه با Google AVD-1-3-3

در طول ایجاد یک AVD، این اختیار را دارید که بین Google device یا Android device یکی را انتخاب نمایید.

یک Android AVD اغلب برنامه هایی از پروژه ی کد باز اندروید (Android Open Source Project) را شامل می شود. این در حالی است که یک AVD ساخته شده با API های Google معمولاً کدهای اضافی (کتابخانه ها یا توابع اضافی) از Google را دربرمی گیرد.

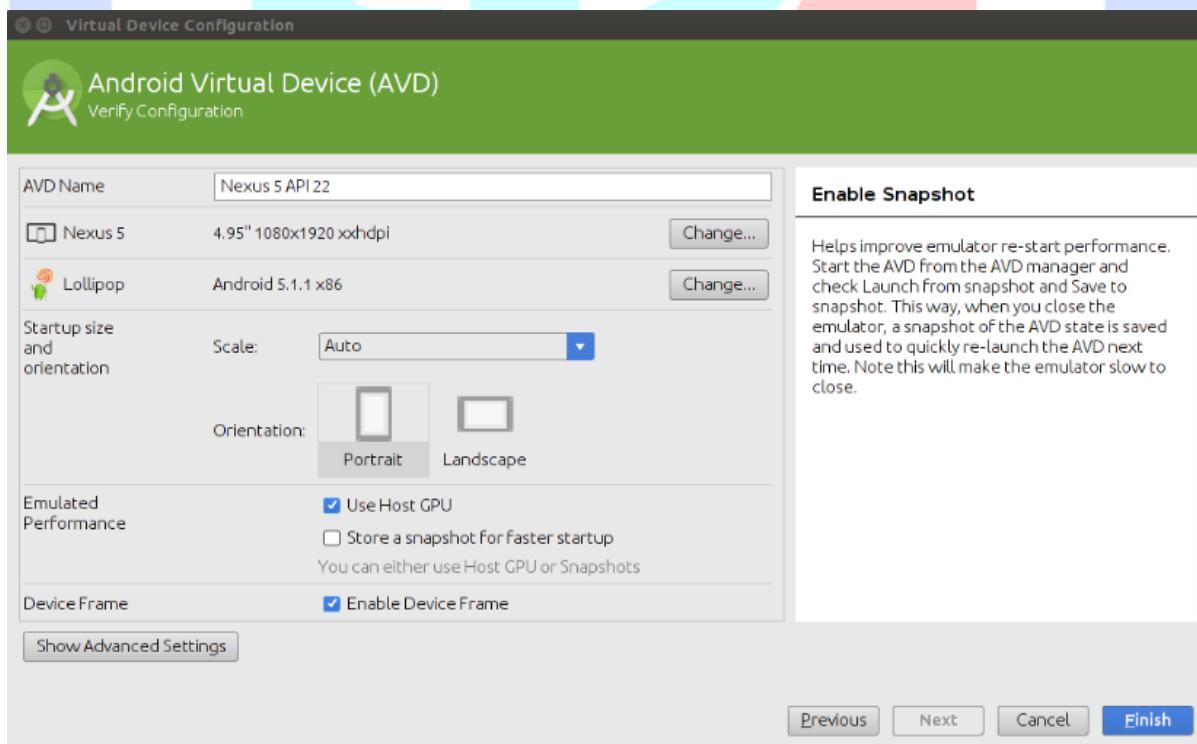
آیینه ای که برای API های گوگل ایجاد می کنید به شما امکان می دهد تا اپلیکیشن هایی را که از سرویس های Google Play (به عنوان مثال سرویس های موقعیت یابی مانند Google Maps API) استفاده می کنند، تست نمایید.

### 4-3-1- گزینه سازی و افزایش سرعت اجرا با انتخاب GPU رایانه میزبان (Emulated Performance)

در طول فرایند تنظیم و آماده سازی یک محیط شبیه ساز برای تست برنامه (emulator)، می توانید یکی از دو گزینه **Snapshot** یا **Use Host GPU** را انتخاب نمایید.

**توجه:** پنجره‌ی محاوره‌ای حاضر به شما اجازه می‌دهد هر دو گزینه را انتخاب نمایید. اما اگر هر دو گزینه را فعال نمایید، با خطا مواجه خواهید شد.

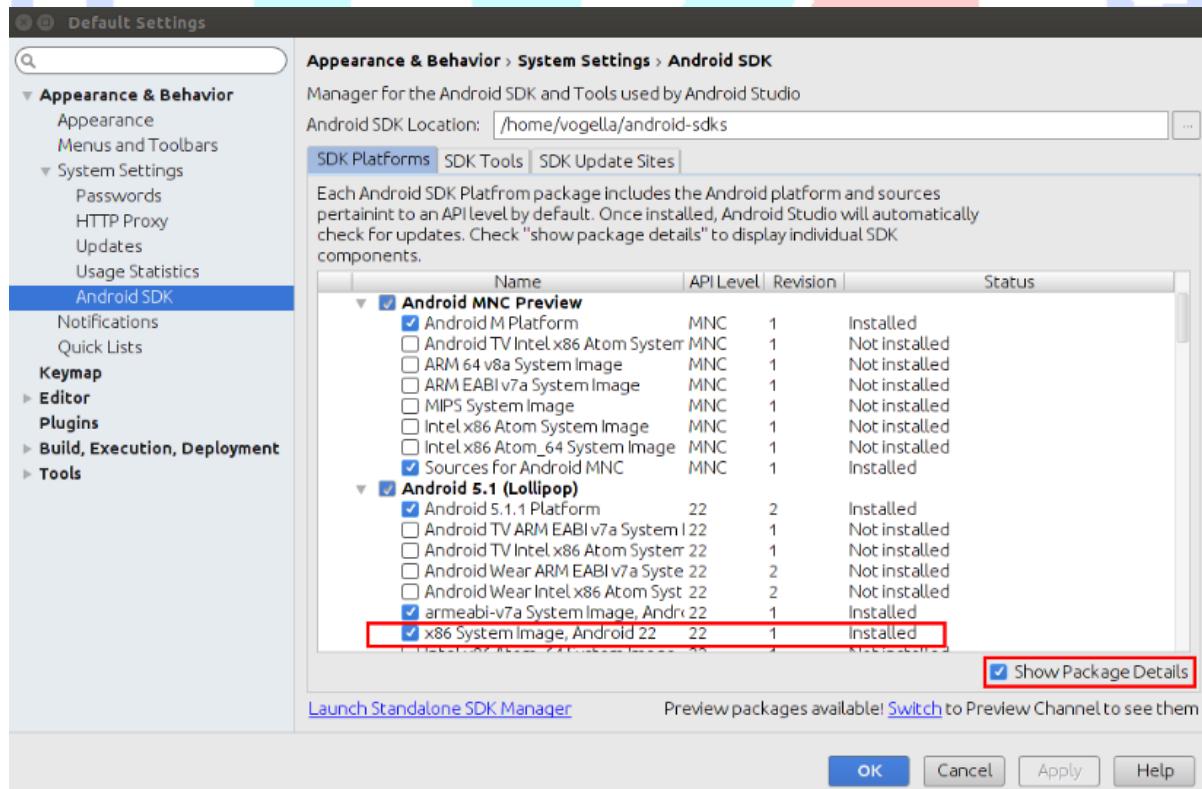
با انتخاب گزینه‌ی اول، هنگامی که شما برای دومین بار دستگاه را راه اندازی می‌کنید، می‌بینید که دستگاه مجازی با سرعت خارق العاده‌ای اجرا می‌شود. می‌دانید چرا؟ به این علت که به هنگام بستن دستگاه مجازی، **AVD** وضعیت و اطلاعات جاری خود را ذخیره می‌کند و طبیعتاً با اجرای بعدی این وضعیت بلافاصله بازگردانده می‌شوند. چنانچه گزینه‌ی **Use Host GPU** را انتخاب نمایید، **AVD** مستقیماً از کارت گرافیک سیستم میزبان برای پردازش و نمایش گرافیک بهره می‌گیرد که سرعت برنامه را در شبیه ساز به طور قابل توجهی بالا می‌برد.



### 1-3-5- افزایش سرعت با انتخاب Intel image system

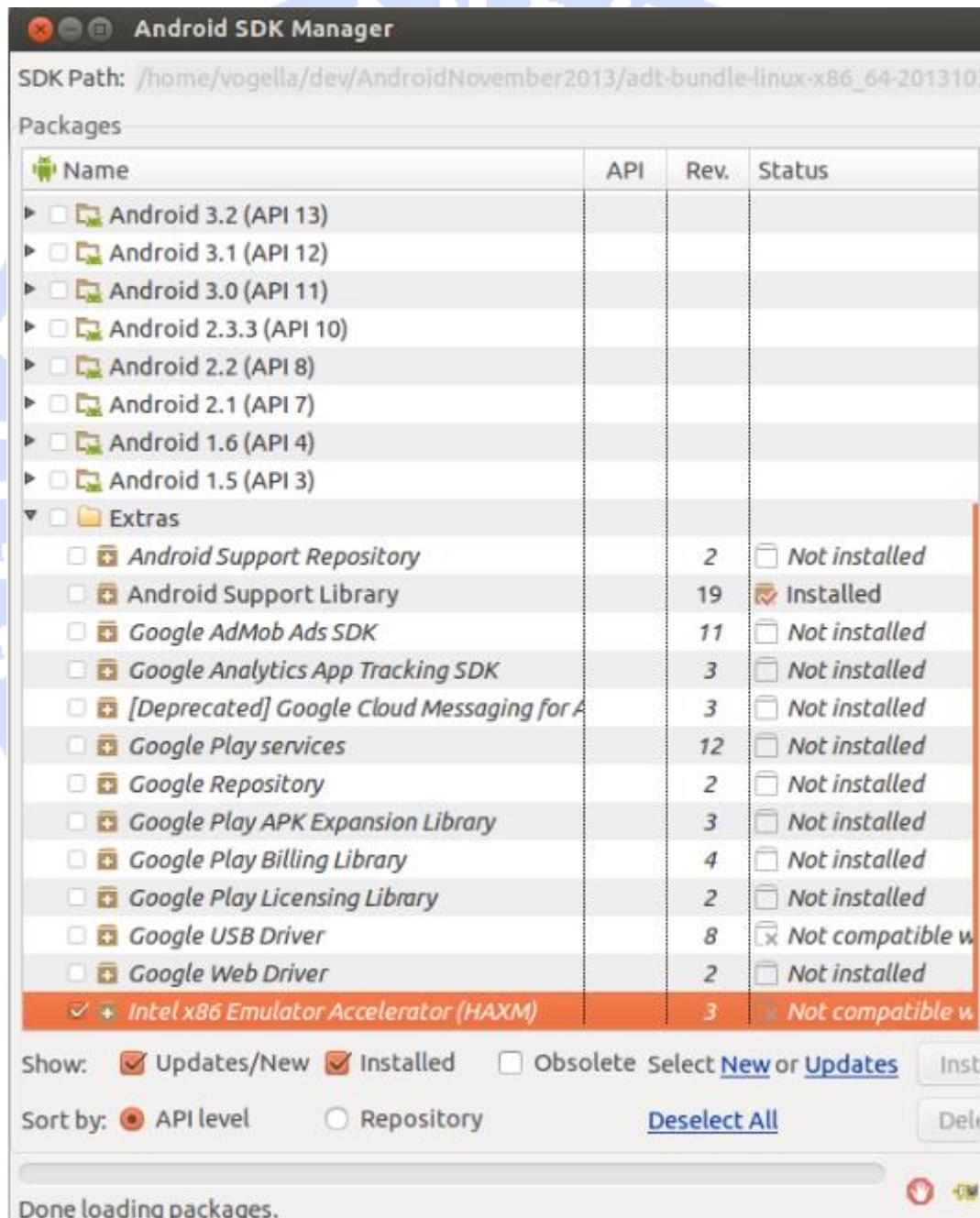
می توانید یک AVD را با image system مبتنی بر معماری ARM CPU یا Intel CPI اجرا نمایید. دستگاه مجازی که از Intel image system بهره می گیرد، در اجرا بر روی سخت افزار AMD/Intel اجرا شود. در اینجا بسیار سریع تر عمل می کند. در شرح علت آن باید گفت که شبیه ساز نیازی ندارد که دستورات پردازنده ی ARM را بر روی کامپیوتر شما به Intel/AMD ترجمه کند. حقیقت این است که image system مبتنی بر ARM بیشتر با دستگاه های اندروید مطابقت دارد، اما لازمه ی این است که پردازنده شبیه سازی شود و این افت سرعت را در پی دارد. بسیار سریع تر است چرا که سعی می کند با استفاده از مکانیزم های مربوطه ی AVD کد را به صورت native اجرا کند.

Intel image system را می توان به راحتی توسط API برای Android SDK Manager نصب کرد. در محیط برنامه نویسی Android Studio، به هنگام ایجاد دستگاه مجازی، این اتفاق به صورت خودکار رخ می دهد. می توان تنظیمات لازم را از طریق package details انجام داد.



**نکته:** لازم به ذکر است که intel image برای تمامی ورژن های مجموعه کتابخانه های اندروید در دسترس نیست.

لازم است برای ویندوز درایورهای بیشتری دانلود و نصب نمایید.



پس از دانلود، می توانید به فایل مورد نظر در محل قرار گیری فایل های نصبی اندروید در پوشه `extras/intel` دسترسی داشته باشید. برای نصب درایورها می بایست است فایل اجرایی `.exe` را کلیک نمایید. این مرحله اضافی برای سیستم عامل ویندوز ضروری است. صرفا دانلود درایور کافی نیست و باید آن را نصب نمایید.

پس از دانلود، می توانید یک `AVD` مبتنی بر شبیه ساز `Intel` ایجاد نمایید. شبیه ساز با سرعت چندان بیشتری راه اندازی نمی شود، اما سرعت اجرای خود اپلیکیشن بسیار سریع تر خواهد بود.

### ۶-۱- تست برنامه بر روی یک دستگاه واقعی اندروید

ابتدا در دستگاه اندروید خود به بخش `setting` مراجعه نموده و در آنجا قابلیت `USB Debugging`

- `USB-Debugging` را فعال نمایید. برای این منظور مسیر رو به رو را طی کنید: `Settings > Development Options`

شاید لازم باشد درایور مورد نیاز برای گوشی همراه خود را نیز نصب نمایید. برای سیستم عامل های `Mac OS X` و `Linux` این درایور از پیش نصب شده و آماده (`out of box`) در اختیار شما قرار می گیرد، اما برای سیستم عامل `Windows` شما می بایست درایور مربوطه را خود نصب نمایید.

**توجه:** پایین ترین ورژن اندرویدی که اپلیکیشن شما بر روی آن قابلیت اجرا دارد و به اصطلاح پشتیبانی می کند، می بایست با نسخه ای اندرویدی که بر روی دستگاه شما نصب است، سازگاری و هم خوانی داشته باشد.

زمانی که چندین دستگاه به کامپیوتر متصل کرده اید، می توانید مشخص کنید کدام یک مورد استفاده قرار گیرد. چنانچه تنها یک دستگاه به کامپیوتر وصل است، اپلیکیشن به صورت خودکار بر روی دستگاه مربوطه مستقر و نصب (`deploy`) می شود.

## ۴-۱- کامپوننت های (اجزای تشکیل دهنده) نرم افزاری یک اپلیکیشن اندروید

## 1-4-1-اپلیکیشن اندروید

اپلیکیشن اندروید در حقیقت یک موجودیت واحد و قابل نصب است که می‌توان آن را راه اندازی نموده و مستقل از دیگر اپلیکیشن‌های اندورید مورد استفاده قرار داد. اجزای تشکیل دهنده یا به اصطلاح کامپوننت‌های نرم افزاری دو اپلیکیشن می‌توانند بر اساس یک آبجکت به نام Intent به یکدیگر وصل شوند. به تعبیر دیگر با استفاده از آبجکت Intent می‌توان عملیات یا task‌هایی تعریف کرده که بین چندین اپلیکیشن رفت و آمد دارند و اطلاعاتی را بین این دو منتقل می‌کنند.

اپلیکیشن‌های اندروید به طور کلی از کامپوننت‌های نرم افزاری، فایل‌های حاوی دستورات Java (source files) و فایل‌های محتوا (resource files) تشکیل می‌شوند. کامپوننت‌های نرم افزاری اندروید هر یک به همراه شرح چیستی و کاربرد آن در زیر عنوان شده‌اند:

1. کامپوننت Application: هر برنامه‌ی اندروید می‌تواند تنها یک کلاس Application داشته باشد که این کلاس قبل از دیگر کامپوننت‌های اندروید نمونه سازی می‌شود (یک نمونه یا آبجکت از این کلاس ساخته می‌شود). این جزء نرم افزاری همچنین آخرین کامپوننتی است که به هنگام خروج و بسته شدن برنامه، متوقف شده و چرخه‌ی حیات آن به اتمام می‌رسد.

اگر خودتان به صورت صریح یک آبجکت از این کلاس نسازید، اندروید خود به صورت خودکار یکی برای شما ایجاد می‌کند.

2. Activity: جز نرم افزاری که یک صفحه از اپلیکیشن را ارائه داده و کاربر با تعامل با آن فعل خاصی را انجام می‌دهد، برای مثال با یکی از مخاطبین تماس گرفته، از منظره‌ای عکس می‌گیرد، یک نقشه را بر روی نمایشگر گوشی خود مشاهده می‌کند یا ایمیلی را ارسال می‌کند. به هر activity یک پنجره اختصاص می‌یابد که در آن رابط کاربری و ظاهر برنامه (UI) ترسیم می‌گردد. به عبارت دیگر، Activity جزء اصلی اپلیکیشن‌های اندروید است که از دو بخش تشکیل شده: 1. کلاس که کد و رفتار برنامه را در آن پیاده سازی می‌کنید 2. لایه‌ی گرافیکی. یک اپلیکیشن می‌تواند چندین activity داشته باشد. activity‌ها با

استفاده از view ها و fragment ها ظاهر برنامه‌ی خود را ساخته و با کاربر تعامل برقرار می‌کنند.

3. Service: یکی از اجزا تشکیل دهنده‌ی اپلیکیشن‌های اندروید که عملیاتی را انجام می‌دهد ولی رابط کاربری یا UI ندارد. به عبارت دیگر، یک کامپوننت نرم افزاری که در پس زمینه کار می‌کند و با کاربر هیچ تعاملی ندارد. سرویس‌ها می‌توانند با دیگر کامپوننت‌های اندروید ارتباط برقرار کنند. به عنوان مثال broadcast receiver می‌تواند از طریق notification framework به کاربر درباره‌ی رخداد خاصی خبر بدهد.

4. Broadcast receiver: یک Broadcast receiver می‌تواند به پیغام‌ها سیستمی و intent ها گوش فرا دهد و به اصطلاح منتظر باشد اتفاق خاصی رخ دهد. در واقع زمانی که مورد نظر رخ می‌دهد، receiver ای که به آن گوش فرا می‌دهد توسط سیستم اندروید مطلع مطلع می‌گردد.

به عنوان مثال می‌توانید یک receiver تخصیص دهید که کار آن گوش دادن به پیغام‌های سیستمی یا intent ها است. یا receiver ای اعلان کنید که منتظر تغییر در وضعیت گوشی (زمانی که گوشی زنگ می‌زند) است.

به عبارت دیگر سیستم اندروید رخدادهایی را که اتفاق می‌افتد توسط broadcast اعلان می‌کند و شما می‌توانید برای این رخدادها receiver تعریف کنید که به رخداد گوش داده و در زمان فعال شدن رخداد از آن مطلع شود.

شما می‌توانید ویژه‌ی event مربوط به اتمام فرایند راه اندازی سیستم (boot)، یک receiver تعریف نمایید.

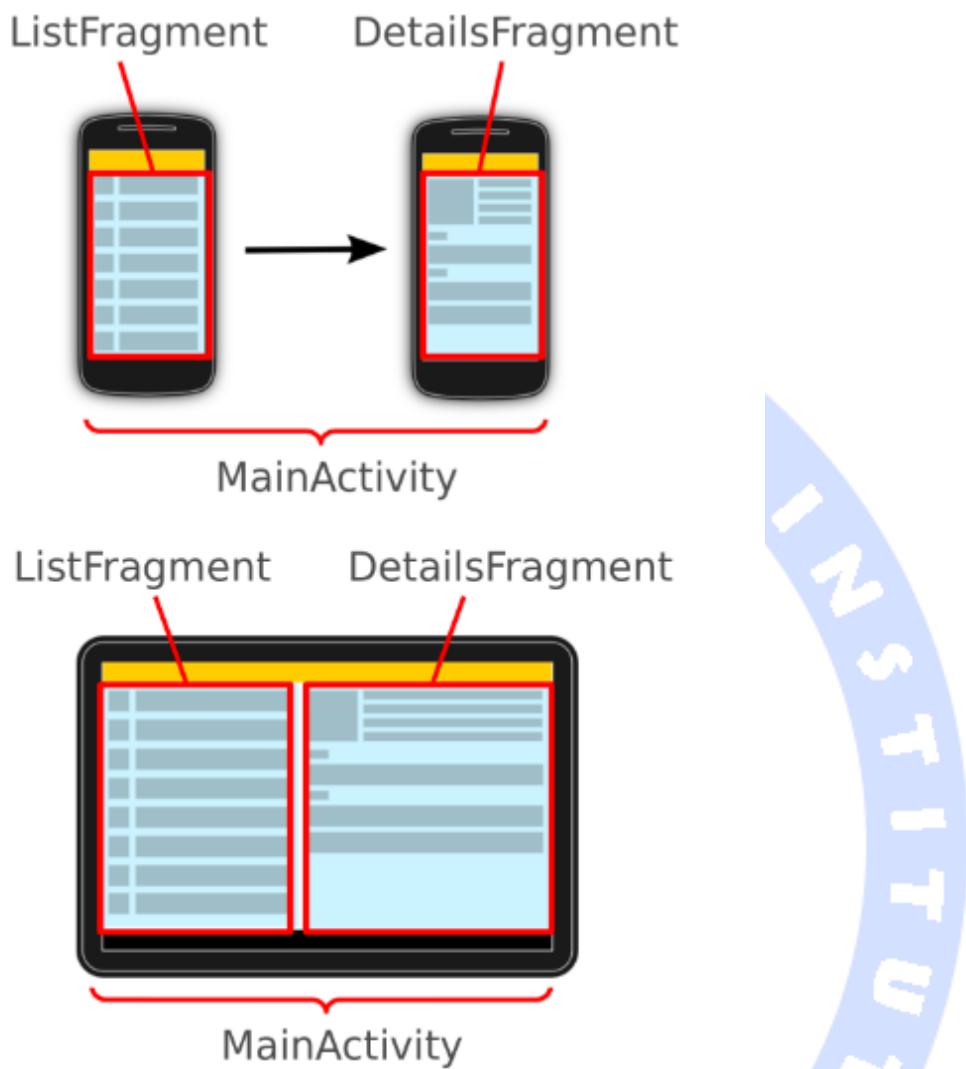
5. Content provider: یک جزء نرم افزاری که رابط ساخت یافته یا یک پل ارتباطی به داده‌های اپلیکیشن فراهم می‌آورد. کامپوننت Content provider این قابلیت را دارد که داده‌های مورد نیاز چندین اپلیکیشن را در مکانی واحد به اشتراک گذاشته تا برنامه‌هایی که به این داده‌ها نیاز دارند بتوانند به آن داده‌ها دسترسی داشته و از آن‌ها استفاده کنند.

سیستم اندروید یک دیتابیس به نام SQLite دارد که به طور مکرر همراه با content provider استفاده قرار می‌گیرد. دیتابیس داده‌ها را ذخیره می‌کند و سپس این داده‌ها توسط واسطهٔ provider در اختیار اپلیکیشن قرار می‌گیرد.

## ۱-۴-۲- طراحی ظاهر و UI برنامه با استفاده از fragment‌ها، view‌ها و layout manager‌ها

Fragment‌ها کامپوننت‌هایی هستند که در بستر (context) یک activity که همان فرم در اندروید است، اجرا می‌شوند. fragment‌ها با کپسوله سازی کدهای اپلیکیشن در خود امکان استفادهٔ مجدد از آن‌ها در activity‌های مختلف را فراهم آورده و همچنین توسعه دهنده را قادر می‌سازد برای دستگاه‌های اندروید با اندازهٔ صفحهٔ متفاوت UI داینامیک و انعطاف‌پذیر تعریف کنند. تصویر زیر یک activity به نام MainActivity را به نمایش می‌گذارد. Activity حاضر، در نمایشگر کوچک، تنها یک fragment را برای کاربر نمایش می‌دهد. این در حالی است که همین activity به هنگام نمایش در دستگاه تبلت هر دو fragment را همزمان در یک صفحه به نمایش می‌گذارد.

آموزشگاه تخلیکر داده



View ها ابزارک های رابط کاربری (UI Widget) همچون دکمه یا text field هستند که به وسیله attribute می توان رفتار و ظاهرشان را تنظیم و ویرایش کرد.

در اندروید مفهومی به نام ViewGroup وجود دارد. ViewGroup خود یک view است که نقش میزبان یا ظرف را برای دیگر view ها (که در اصطلاح view های فرزند آن خوانده می شوند) ایفا می کند. به این view میزبان در اندروید layout manager نیز گفته می شود چرا که قادر است چینش view های دیگر را مدیریت کند.

کلاس پایه که از آن ارث بری می کند android.view.ViewGroup است. این کلاس خود از android.view.View که کلاس پدر تمامی view ها است مشتق می شود.

می توان با قرار دادن layout manager ها در دل یکدیگر (تودرتو کردن آن ها)، layout یا قالب های ترکیبی و پیچیده تری ایجاد کرد.

### 3-4-1-ابزارک های رابط کاربری یا widget های مورد استفاده در صفحه اصلی (home screen widget)

Widget ها یا ابزارک های رابط کاربری که در صفحه اصلی موبایل به نمایش در می آیند، در واقع یک سری broadcast receiver هستند که کامپوننت های تعاملی ارائه می دهند. شما از طریق این کامپوننت ها می توانید به بخش هایی از برنامه در صفحه اصلی موبایل (home screen) دسترسی داشته باشید. به عنوان مثال، یک ابزارک می تواند به کاربر اجازه دهد خلاصه ای از ایمیل های جدید را مشاهده کند و زمانی که کاربر بر روی ایمیل دلخواه خود کلیک کرد، کاربر را به صفحه مربوطه در اپلیکیشن هدایت کند.

Live wallpapers به شما این امکان را می دهند تا background های متحرک و پویا برای نمایش در صفحه اصلی سایت خود ایجاد کنید.

### 3-4-2-کلاس Context

نمونه یا آبجکت های ایجاد شده از کلاس android.content.Context، اتصال/امکان دسترسی به سیستم اندروید و دستگاهی که اپلیکیشن بر روی آن اجرا می شود را فراهم می آورد. به عبارت بهتر، نمونه های این کلاس به سیستم، منابع و سرویس های اپلیکیشن دسترسی دارند.

برای مثال، می توانید به وسیله ای Context اندازه دقیق صفحه نمایش دستگاه جاری را بدست بیاورید.

Activity ها و سرویس ها از کلاس Context ارث بری می کنند.

### 5-1-فایل تنظیمات اندروید (manifest)

## ۱-۵-۱- تنظیمات اپلیکیشن های اندروید

کامپوننت ها، تنظیمات و metadata یک اپلیکیشن اندروید در فایل AndroidManifest.xml کامپوننت آن که فایل تنظیمات در سیستم اندروید است، قید می شود.

تمامی activity ها، سرویس ها و کامپوننت های content provider اپلیکیشن باید در این فایل و به صورت static تعریف شوند. اما Broadcast receiver را می توان به صورت static در فایل manifest یا به صورت dynamic و در زمان اجرا در اپلیکیشن تعریف کرد. سیستم اندروید این فایل تنظیمات را به هنگام نصب اپلیکیشن خوانده و سپس با توجه به اطلاعات داخل آن، قابلیت ها و امکانات متعدد اپلیکیشن مورد نظر را شناسایی می کند.

Gradle، سیستم کامپایل و دسته بندی کلاس های اندروید تحت یک پوشه ی واحد (build&packaging)، می تواند فایل manifest را ایجاد کرده و در اختیار شما قرار دهد. برای مثال، ورژن اپلیکیشن معمولاً توسط فایل gradle build عرضه می شود.

## ۱-۵-۲- نمونه ای از فایل تنظیمات اندروید (manifest)

در زیر نمونه ای از یک فایل ساده‌ی تنظیمات اندروید (manifest) را مشاهده می کنید:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.android.rssreader"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk
        android:minSdkVersion="16"
        android:targetSdkVersion="19" />
    <uses-permission android:name="android.permission.INTERNET" />
    <application
        android:name="RssApplication"
        android:allowBackup="false"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="RssfeedActivity"
            android:label="@string/title_activity_main" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
```

```

<activity
    android:name=".DetailActivity"
    android:label="Details" >
</activity>
<activity android:name="MyPreferenceActivity" >
</activity>
<service android:name="RssDownloadService" >
</service>
</application>
</manifest>

```

### 3-5-1- خصیصه های Package و تعین نسخه های اپلیکیشن با استفاده از **manifest** در فایل تنظیمات

در فایل تنظیمات attribute یا خصیصه ای به نام package مشاهده می شود که اسم پکیجی که آبجکت ها (کلاس های) جاوا مورد اشاره در این فایل، داخل آن قرار دارند را مشخص می کند. در صورتی که یک آبجکت جاوا داخل پکیج دیگری قرار دارد، در آن صورت لازم است اسم کامل پکیج قید شود.

برای عرضه ی اپلیکیشن در فروشگاه مجازی Google play، آن اپلیکیشن باید یک اسم پکیج اختصاصی و منحصر بفرد داشته باشد. توصیه می شود برای اسم پکیج، اسم دامنه ی اپلیکیشن خود را به صورت وارونه بکار ببرید. بدین وسیله یک معرف یکتا برای اپلیکیشن اعلان می کنید که آن را از دیگر اپلیکیشن ها برای سیستم اندروید تمیز می دهد.

مقدار **versionCode** و **versionName** اپلیکیشن را مشخص می کنند. **versionCode** این مقدار قابل مشاهده برای کاربرانی است که اپلیکیشن را بارگیری و نصب می کنند و هر می تواند هر مقدار رشته ای باشد.

مقدار **versionCode** باید یک عدد صحیح یا integer باشد. به وسیله ی این مقدار عددی Google play هم خود اپلیکیشن را شناسایی می کند و هم نسخه های جدید و آپدیت های آن را دنبال می کند. مقدار این خصیصه معمولاً در اولین ورژن بر روی 1 تنظیم می شود و همین طور که ویرایش های جدید از آن منتشر می شود، این مقدار بالا می رود.

## ۱-۵-۴-تگ <application> (یک ظرف برای اجزا) و کامپوننت های دیگر

به شما این امکان را می دهد تا برای اپلیکیشن خود metadata بنویسید (اطلاعاتی را پیرامون برنامه ی خود ارائه دهید) و علاوه بر آن یک کلاس application به صورت صریح تعریف نمایید. این تگ همچنین می تواند ظرفی برای تعریف دیگر کامپوننت های اندروید باشد.

توسط تگ <activity> یک کلاس activity تعریف می شود. در این تگ خصیصه (attribute) name به کلاسی اشاره دارد که اگر به طور کامل قید نشده باشد، نسبت به پکیجی که در خصیصه package آن اپلیکیشن قید کردید، آدرسی دهی می شود.

در واقع </action android:name="android.intent.action.MAIN" /> بیانگر این است که اپلیکیشن می تواند با این activity اجرا شود و این activity نقطه ی اجرا برنامه می باشد. پارامتر <category android:name="android.intent.category.LAUNCHER" /> نیز اعلان می دارد که حاضر بایستی به لانچر اضافه شود (در لیست اپلیکیشن های دستگاه به صورت فایل اجرایی ارائه شود).

مقدار <resource @string/app\_name> به فایل های محتوا (resource) اشاره دارد که اسم اپلیکیشن در آن تعریف شده است. با بهره گیری از فایل های محتوا می توان به راحتی منابع، محتوای مختلف (مقادیر رشته ای، رنگ و آیکون برنامه) و مورد نیاز هر اپلیکیشن را فراهم نموده و همچنین زبان برنامه را به راحتی تنظیم کرد (اطلاعات مربوط به زبان برنامه را فراهم و آن را ترجمه کرد).

تگ receiver و provider نیز می توانند مانند activity، نقش ظرف را برای تعریف دیگر کامپوننت و اجزا اندروید ایفا کنند.

## ۵-۵-۱- تعیین پایین/بالاترین نسخه اندروید که برنامه بر روی آن اجرا می شود (خصوصیه های targetSdkVersion و minSdkVersion)

بخش uses-sdk در فایل تنظیمات به شما این امکان را می دهد که پایین ترین نسخه و همچنین نسخه ای مورد نظر از اندروید که می خواهید اپلیکیشن بر روی آن قابل اجرا باشد را به ترتیب از طریق targetSdkVersion و minSdkVersion attribute مشخص نمایید.

.1: این خصیصه به شما امکان می دهد پایین ترین ویرایش اندروید که اپلیکیشن شما بر روی آن قابل اجرا می باشد را مشخص نمایید. مقداری که اینجا مشخص می کنید در واقع به منزله ی یک فیلتر در فروشگاه مجازی Google Play مورد استفاده قرار می گیرد، به طوری که دستگاه هایی که اندروید نصب شده بر روی آن پایین تر از مقدار مشخص شده در این attribute است، امکان نصب آن را پیدا نمی کنند.

.2: مقدار این targetSdkVersion attribute نشانگر ورزشی از سیستم عامل اندروید است که اپلیکیشن در اصل ویژه ی آن طراحی و بر روی آن تست شده است. لازم به ذکر است که این مقدار برابر ورزش کتابخانه های اندروید یا API version دستگاه نیست. سیستم اندروید ممکن است جهت سازگاری و قابلیت اجرای برنامه بر روی ورزش های جدید یا قدیمی تر تغییراتی را اعمال کند (forward/backward compatibility). توصیه می شود این attribute را جهت بهره وری از آخرین امکانات اندروید، برابر آخرین API version قرار دهید.

## ۵-۵-۲- تعیین مجوزهای دسترسی با استفاده از تگ permission

فایل تنظیمات اندروید (manifest) همچنین می بایست مجوزهای لازم را مشخص کند. به عنوان مثال، در صورتی که اپلیکیشن به اینترنت نیاز دارد، مجوز دسترسی به آن باید در این فایل تعریف و اعطای شود.

اپلیکیشن با استفاده از تگ <uses-permission> به سیستم اندروید اعلان می کند که به مجوز خاصی احتیاج دارد.

برخی از مجوزها، همچون دسترسی به اینترنت، از اندروید 6.0 به بعد به صورت خودکار اعطا می شوند. اما برخی دیگر جهت فعال شدن به تاییدیه ی کاربر نیاز دارند.

## 7-5-1- تعیین سیستم سخت افزاری مورد نیاز (بخش uses-configuration و uses-feature در فایل تنظیمات)

در فایل تنظیمات (manifest) اندروید همچنین بخشی به نام uses-configuration وجود دارد که روش های دریافت ورودی برای دستگاه را مشخص می کند. برای مثال، تکه کد زیر اعلان می کند که دستگاه مورد نظر بایستی از صفحه کلید مجزا (به صورت سخت افزاری) برخوردار باشد.

```
<uses-configuration android:reqHardKeyboard="true"/>
```

بخش uses-feature به شما این امکان را می دهد تا سیستم سخت افزاری که دستگاه میزبان اپلیکیشن باید داشته باشد را مشخص نمایید. به عنوان مثال، تکه کد زیر اعلان می کند که دستگاه مورد نظر باید برای اجرای اپلیکیشن و بهره وری از قابلیت های آن، دوربین داشته باشد.

```
<uses-feature android:name="android.hardware.camera" />
```

## 7-5-1- تعیین محل نصب (خصوصیه ی installLocation)

به واسطه ی خصوصیه ی installLocation می توانید مشخص کنید آیا اپلیکیشن شما اجازه یا امکان نصب بر روی حافظه ی خارجی دستگاه را دارد یا خیر. برای اعطای این مجوز کافی است ذکر شده را با یکی از دو مقدار auto یا preferExternal قرار دهید.

در حقیقت، این امکان به ندرت مورد استفاده قرار می گیرد چرا که اپلیکیشن (مسقر بر روی حافظه ی خارجی) به مجرد وصل شدن به کامپیوتر و استفاده از آن به عنوان یک حافظه ی USB، بلافاصله متوقف می شود.

## 1-6-فایل های محتوا (Resources)

اندروید به شما اجازه می دهد فایل های محتوا پروژه را به صورت مجزا و جدا از کد برنامه در قالب منابع static (فایل های تنظیمات مبتنی بر XML ، عکس، متن و غیره ...) تعریف نمایید.

فایل های محتوا باستنی داخل پوشه `/res` / اپلیکیشن شما، در زیرپوشه ای از پیش تعریف شده جایگذاری شود. این زیرپوشه معمولاً به نوع محتوایی که در آن قرار داده می شود، بستگی دارد. در صورت لزوم می توانید شناسه های بیشتری به اسم پوشه الصاق نموده و از این طریق مشخص نمایید محتوا مربوطه باید برای تنظیمات خاصی مورد استفاده قرار گیرد. این شناسه ها در اصطلاح تنظیم محتوا مربوطه خوانده می شوند. به عنوان مثال، می توانید مشخص کنید که فایل `layout` یا `resource qualifier` تنظیم چیدمان های المان رابط کاربری فقط برای اندازه صفحه خاصی قابل استفاده می باشد.

جدول زیر توضیح مختصری در خصوص منابع (resource) مورد پشتیبانی و پیشوند مربوطه هر یک ارائه می دهد.

Resource یا محتوا مورد نظر	پوشه های مربوطه	شرح
Drawables	<code>/res/drawables</code>	فایل های تصویری (برای مثال png، drawable/jpeg)، فایل های XML که به طور خودکار خود را متناسب با تراکم پیکسلی نمایشگر اندازه بندی و تنظیم می کند را دربرمی گیرد. می توان از طریق کلاس R.drawable به آن ها دسترسی داشت.
Simple Values	<code>/res/values</code>	داخل این فایل مقادیر متنی یا رشته ای، رنگ ها، ابعاد و اندازه، استایل ها و آرایه ای از رشته ها یا اعداد صحیح در قالب فایل XML تعریف و ذخیره می

محتوای مورد نظر یا Resource	پوشه ی مربوطه	شرح
		شوند. به طور قرار دادی، هر نوع در پوشه ی جدا و مختص به خود قرار می گیرد، برای مثال مقادیر رشته ای در آدرس <code>res/values/strings.xml</code> ذخیره می شوند.
Layouts	<code>/res/layout</code>	فایل های XML با مقادیر مربوط به <code>layout</code> اپلیکیشن که امکان طراحی محیط کاربری با <code>activity</code> ها و <code>fragment</code> ها را فراهم می آورند در این مسیر نگهداری می شوند.
Styles and themes	<code>/res/values</code>	فایل هایی که ظاهر و سبک نمایش اپلیکیشن اندروید را تعریف می کنند.
Animations	<code>/res/animation</code>	می توانید دستورات انیمیشن را به صورت xml داخل این فایل (برای کار با <code>animation API</code> ) تعریف کنید که به <code>property</code> های آبجکت را جهت اجرای انیمیشن و ادامه ی آن تا مدت زمان خاص، ویرایش نمایید. به عبارت دیگر فایل های ویژگی های انیمیشن را تعریف می کنند، در این مسیر ذخیره می شوند.
Raw data	<code>/res/raw</code>	داده هایی که می خواهید با فرمت خام خود ذخیره شوند را در این فایل تعریف می کنید. می توانید از طریق آبجکت <code>InputStream</code> به راحتی به آن ها دسترسی داشته باشید. در واقع بهتر است فایل های که فرمت آن ها با دیگر فایل متفاوت است در این پوشه جایگذاری نمایید.

محتوای مورد نظر یا Resource	پوشه ی مربوطه	شرح
Menus	/res/menu	فایل های XML ای که منوهای اپلیکیشن را تشکیل می دهند در این پوشه قرار دارند. برای دسترسی به آن ها کافی است از R.menu استفاده نمایید.

## ۱-۶-۱-نمونه فایل resource (تعریف تعدادی ثابت رشته ای، آرایه ی رشته ای، ثابت رنگ و ابعاد)

به یک نمونه فایل resource می توان به values.xml در پوشه ی /res/values اشاره کرد. این فایل دربردارنده ی تعدادی ثابت رشته ای، آرایه ی از نوع رشته، ثابتی حاوی مقدار رنگ و ثابتی حاوی مقدار ابعاد می باشد.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Test</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>
    <string-array name="operationsystems">
        <item>Ubuntu</item>
        <item>Android</item>
        <item>Microsoft Windows</item>
    </string-array>
    <color name="red">#ffff0000</color>
    <dimen name="mymargin">10dp</dimen>
</resources>
```

## ۱-۶-۲-فایل های R.java و resource

سیستم build و کامپایل اندروید به تمامی فایل های محتوا و resource های مربوطه در پوشه ی res، یک ID منحصر بفرد تخصیص می دهد. اندروید سپس یک فایل به نام R.java ایجاد می کند که مقادیر این ID ها را در خود ذخیره می کند. ID ها (اشارة گرها) در واقع مقادیر static و عددی از نوع int هستند.

بلافاصله پس از ایجاد یک فایل resource جدید، ID متناظر با آن به صورت خودکار در فایل R.java ایجاد می شود. نیازی به تغییرات دستی در فایل R.java نیست و در صورت اعمال چنین تغییراتی هم، Android development tooling آن ها را بازنویسی می کند. سیستم اندروید تعدادی متداهن می دهد که ID مربوط به یک resource را به عنوان آرگومان پذیرفته و فایل resource متناظر آن را بازیابی می کند.

برای مثال، جهت دسترسی به یک String با ID منحصر بفرد R.string.yourString در کد برنامه، شما می بایست متداهن Context را فراخوانی نمایید.

### ۱-۶-۳- فایل های چیدمان رابط کاربری (Layout)

Activity ها (یا فرم ها در اندروید) ظاهر و UI خود را با استفاده از view ها (widget یا ابزارک های رابط کاربری) و fragment ها می سازند. در سیستم عامل اندروید کدهای مربوط به چیدمان المان های UI و طراحی رابط کاربری در قالب فایل های محتوا مبتنی بر XML در پوششی ذخیره می گردند. در صورت تمایل می توانید تلفیقی از دو روش را بکار ببرید، هر چند روش اول بیشتر توصیه می شود. بدین وسیله می توان منطق برنامه (programming logic) را از ظاهر آن (layout definition) جدا کرد. همچنین این امکان بوجود می آید که برای هر دستگاه ظاهر و فایل layout ویژه ای تعریف نمایید.

فایل محتوا (resource file) که کدهای UI برنامه از آن خوانده می شوند در اصطلاح برنامه نویسی اندروید layout گفته می شود. Layout یا فایل تنظیم چیدمان المان های رابط کاربری تمامی ViewGroup ها، view ها و رابطه میان آن ها و attribute های آن ها را در قالب فایل های XML تعریف می کنند.

کد زیر یک نمونه ساده از فایل layout را به نمایش می گذارد.

```
<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"
    android:paddingBottom="@dimen/activity_vertical_margin"
```

```

    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >
    <TextView
        android:id="@+id/mytext"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />
</RelativeLayout>

```

فایل layout با فراخوانی متده استفاده می کند تا activity در UI تخصیص می یابد، مانند زیر:

```

package com.vogella.android.first;
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}

```

#### ۱-۶-۴- افزایش سرعت و کارایی با استفاده از layout های ساده

محاسبه، پردازش و ترسیم view ها یک عملیات سنگین بوده و منابع و محتوای حجمی را شامل می شود. شما بایستی تا حد امکان از layout های ساده استفاده نمایید تا سرعت اجرا و کارایی کلی برنامه بهبود یابد. برای مثال، بهتر است در شرایطی که یک layout manager که یک layout manager ساده کفايت می کند، از تودرتو سازی بیش از حد layout manager ها خودداری نمایید.

#### ۱-۶-۵- رهنمودها و روش های بهینه در خصوص ID های اختصاص داده شده به فایل های محتوا/resource

به منظور دسترسی به یک view از کد جاوا یا XML، لازم است با تنظیم مقدار attribute ای به نام ID view، به دلخواه ID یا شناسه‌ی منحصر بفردی تخصیص دهید. جهت تخصیص یک ID جدید به view مورد نظر، کافی است خصیصه‌ی android:id از المان متناظر را در فایل layout با مقدار دلخواه تنظیم نمایید.

camelCase یا مجموعه ابزار ساخت توسعه برنامه های اندرویدی، از سیستم نشانه گذاری SDK برای نگارش مقدار ID بهره می گیرد.

مثال: buttonRefresh. توصیه می شود شما نیز از همین سیستم استفاده نمایید.

در نمونه ی زیر می بینید که با تنظیم مقدار attribute نام برد (خصوصیه ی android:id) بر روی پارامتر رشته ای "@+id/button1"، به المان دکمه یک شناسه ی منحصر بفرد اختصاص داده می شود. طبق قراردادهای تعریف شده، این دستور یک ID جدید در فایل R.java ایجاد کرده و سپس آن را به view مربوطه انتساب می دهد.

```
<Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Show Preferences" >
</Button>
```

توصیه می شود تمامی ID ها را در یک فایل تنظیمات (Config) واحد تعریف نمایید. این فایل معمولاً ids.xml نام گذاری شده و در پوشه ی /res/values/ جایگذاری می شود. با این کار شما می توانید ID های از پیش تعریف شده را در فایل layout مورد استفاده قرار دهید. لازم به ذکر است که شما می توانید ID ها را در فایل جداگانه تعریف نمایید، اما در آن صورت بایستی آیتم @+id را از فایل های layout خود حذف نمایید چرا که در غیر این صورت با یک پیغام خطأ مبنی بر اینکه این فایل ها قبلاً ساخته شده اند مواجه می شوید. کد زیر نمونه ای از این فایل را به نمایش می گذارد.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <item name="button1" type="id"/>
</resources>
<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" >
    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:layout_marginRight="27dp"
        android:text="Button" />
</RelativeLayout>
```

**توجه:** روشی که هم اکنون از آن نام برده شده برای پروژه های واقعی بسیار کارامد و مناسب می باشد. اما در آموزش حاضر، به علت زمان بر بودن این پروسه، مورد استفاده قرار نمی گیرد.

## 1-6-6-محتوا و منابعی که توسط سیستم اندروید ارائه می شود (system resources)

سیستم اندروید خود نیز محتوا و resource ارائه می دهد. به این محتویات و منابع system resources گفته می شود. فایل های resource که خود سیستم در اختیار برنامه نویس قرار می دهد توسط پیشوند android از دیگر فایل های محتوا (local resource) متمایز می شوند. به عنوان مثال، یک string است که محیط اندروید (platform) ویژه‌ی cancel تعیین می کند.

## 1-7-مفهوم view در اندروید - المان ها و ابزارک های رابط کاربری یا UI Widget

همان طور که می دانید یک اپلیکیشن اندروید دارای یک یا چند activity هست. همان فرم یا پنجره در برنامه های تحت ویندوز است. حال داخل هر activity شما می توانید المان های رابط کاربری همچون دکمه، کادر متن و غیره داشته باشید. این اجزا رابط کاربری در واقع نمونه هایی از کلاس view یا subclass های (کلاس های ارث بری شده از) ViewGroup هستند.

به عبارت دیگر view کلاس پدر (superclass) است که تمامی کامپوننت های GUI در اندروید از آن ارث بری می شوند. به عنوان مثال، کلاس TextView که یک برچسب متنی را در UI به نمایش می گذارد یک کلاس فرزند از View است (ViewGroup نیز یک کلاس مشتق شده از View است. از خود ViewGroup آبجکت هایی ساخته می شود که این آبجکت ها نقش ظرف را برای گروه بندی نمونه های کلاس View ایفا می کند. به عنوان مثال می توان به کلاس LinearLayout اشاره کرد که یک کلاس فرزند از ViewGroup است).

هر view در اندروید نماینده ی یک widget (ابزارک یا المان کاربری) همچون دکمه یا یک layout سیستم اندروید این ابزارک یا المان های رابط کاربری را از طریق کلاس manager هست. SDK

در اختیار شما قرار می دهد ( ListView که لیستی از آیتم ها را در نمایشگر به صورت فهرست ارائه می دهد ) .

تمامی view ها از کلاس android.view.View ارث برده می شوند. این کلاس پدر، بسیار سنگین و حاوی حدودا 18 هزار خط کد است که قابلیت های پایه ای مختلفی را در اختیار کلاس های فرزند خود قرار می دهد.

کلاس های پایه ی view ها همگی در پوشه ی android.view قرار دارند و widget های پیش فرض محیط اندروید ( platform ) داخل پوشه ی android.widget دارند.

## ViewGroup و layout manager -8-1

### layout manager از 8-1

وظیفه ی تنظیم ظاهر خود و مدیریت چیدمان view های فرزندش را بر عهده دارد. در واقع ViewGroup یک کلاس مشتق شده از Layout manager است.

Layout manager های متعددی در سیستم اندروید تعبیه شده که در زیر به پرکاربردین آن ها اشاره می شود:

- توسط یک کتابخانه ی خارجی به اندروید اضافه می شود.
- ConstraintLayout
- LinearLayout
- FrameLayout
- RelativeLayout
- GridLayout

### 8-2- تنظیم و ویرایش attribute ها از طریق layout manager

تمامی layout manager ها را می توان با ویرایش مقادیر attribute به آسانی طبق نیاز تنظیم کرد. View های موجود در layout manager می توانند هایی داشته باشند و از طریق آن ها ظاهر خود همچون اندازه ی عرض و طول را مشخص کنند.

View های موجود در layout manager می توانند طول و عرض مد نظر خود را به واسطه attribute زیر تنظیم کنند.

- عرض widget را تعیین می کند: android:layout\_width
- طول widget را مشخص می کند: android:layout\_height



اندازه layout manager ها را می توان بر حسب واحد های اندازه گیری متعدد مشخص نمود یا مقادیر آمده و از پیش تعریف شده layout را مورد استفاده قرار داد. مثال: 100dp.

مقدار view layout manager به اپلیکیشن دستور می دهد widget موجود در match\_parent (یا view) را دقیقا از لحاظ اندازه با view پدر (میزان) برابر قرار دهد. در مقابل این پارامتر، مقدار wrap\_content وجود دارد که به layout اعلان می کند تنها به میزان مورد نیاز به widget فضای اختیص دهد. در تصویر زیر هر دو پارامتر به نمایش گذاشته شده است.



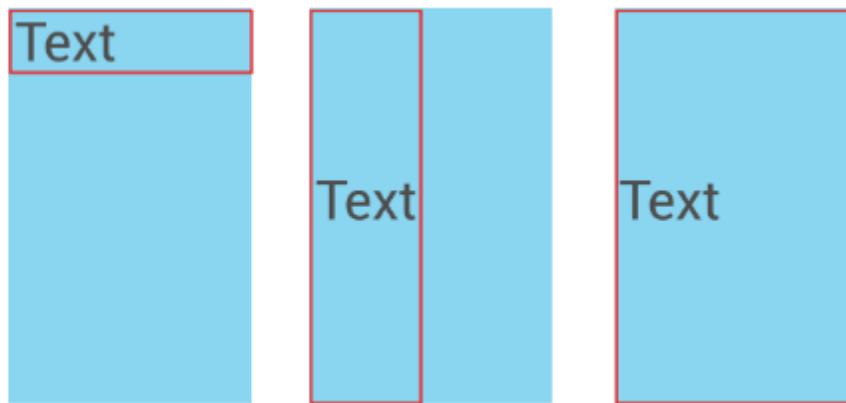
# wrap\_content

```
<TextView  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    ...  
/>
```



# match\_parent

```
    android:layout_width="wrap_content"  
    android:layout_height="match_parent"  
  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"
```



## Constraint Layout-1-8-3

یکی از ابزار کنترل ظاهر و چیدمان المان های UI است که توسط کتابخانه `Constraint layout` خارجی در اختیار توسعه دهنده قرار می گیرد. این نوع `layout` به شما امکان استفاده از `view` `hierarchy` تخت و `flat` را می دهد که در افزایش سرعت اجرای اپلیکیشن بسیار موثر است. علاوه بر آن تمامی ابزارهای طراحی (`design tool`) به راحتی از `constraint layout` پشتیبانی می کنند.

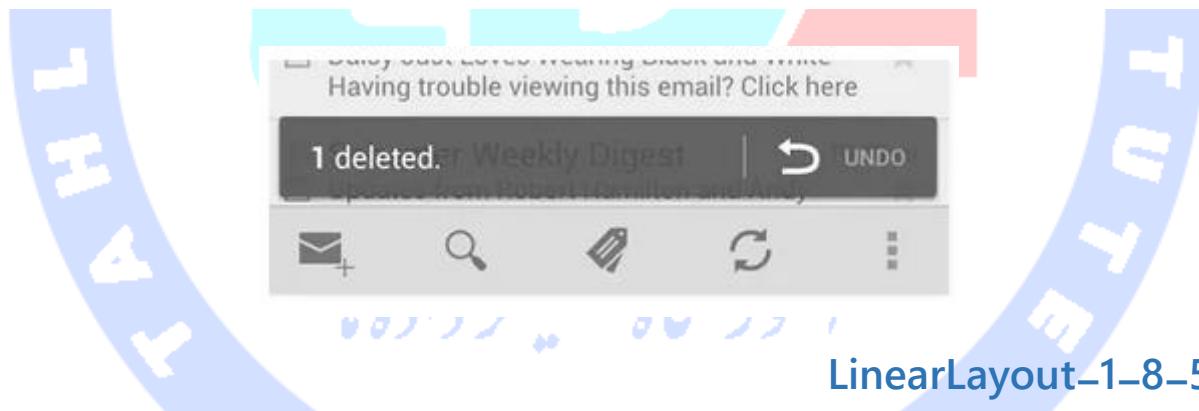
در پروژه های جدید بهتر است که از constraint layout برای طراحی ظاهر اپلیکیشن استفاده شود.

در واقع با constraint layout می توان قالب های انعطاف پذیر، واکنشگرا و پیچیده (تودرتو) تعریف کرد که علاوه بر قابلیت تطبیق خودکار خود با نمایشگرهای مختلف و تغییر جهت نمایش یا وضعیت چیدمان، می تواند سرعت بالایی داشته باشد.

## FrameLayout-1-8-4

یک layout manager FrameLayout است که تمامی المان های داخل خود (child element) را بر روی هم ترسیم می کند. این به شما امکان می دهد جلوه های بصری و افکت های جالب طراحی کنید.

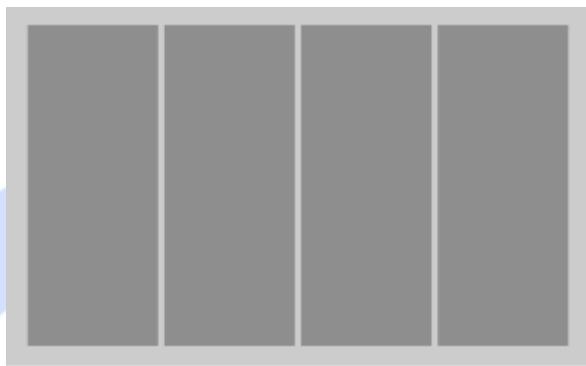
تصویر زیر اپلیکیشن Gmail را به نمایش می گذارد که با استفاده از FrameLayout چندین دکمه را بر روی layout دیگر قرار می دهد. در این layout المان ها می توانند بر روی یکدیگر قرار گیرند.



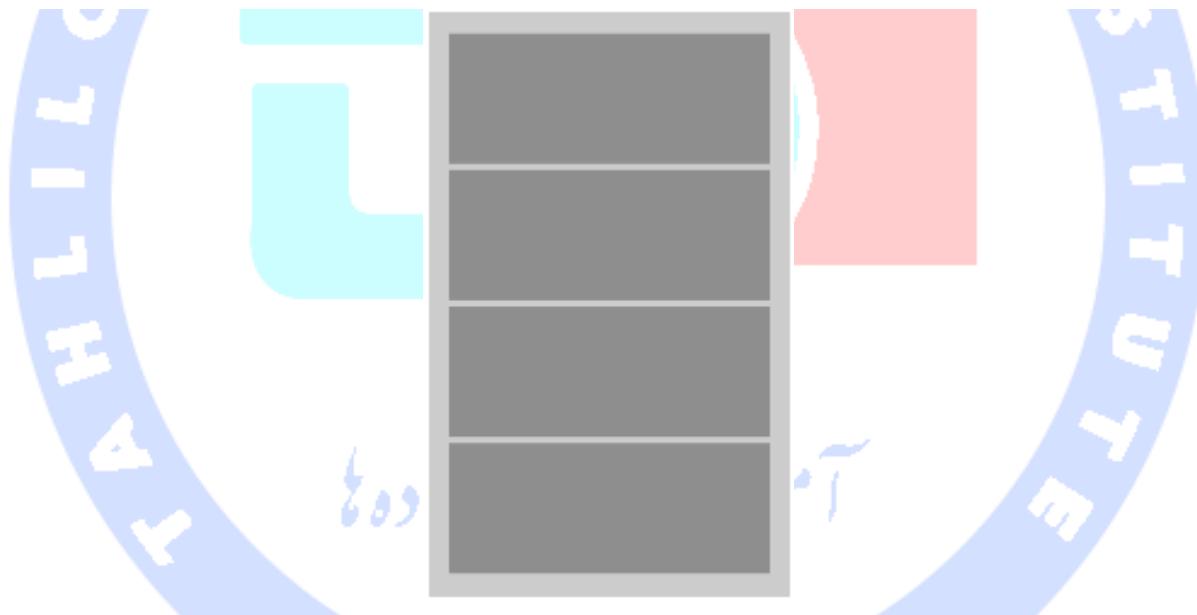
## LinearLayout-1-8-5

تمامی المان های داخل خود را، بر اساس مقدار خصیصه‌ی orientation، دورن یک سطون یا سطر واحد قرار می دهد. در واقع با استفاده از این layout المان های فرزند همگی پشت سرهم در یک مسیر قرار می گیرند. همان طور که گفته شد، این layout یک attribute به نام orientation دارد که می توان آن را با دو مقدار مختلف تنظیم کرد: horizontal و vertical مقدار horizontal که المان مربوطه را به صورت افقی نمایش می دهد، حالت پیش فرض می باشد.

در صورت مقداردهی attribute name بده با `horizontal`, المان های داخل layout manager به صورت زیر در نمایش گر ارائه می شوند.



در صورت تنظیم مقدار attribute name بر روی `vertical`, المان های داخل layout manager به صورت زیر در نمایشگر نشان داده می شود:



را می توان در حالت `nested` و تودرتو بکار برد. به عبارت دیگر این امکان وجود دارد که با تعریف کردن چند layout داخل یکدیگر، قالب های پیچیده و جالب تر ایجاد نمود. در `LinearLayout` شما می توانید اندازه هر یک از المان های حاضر در layout manager را با مقداردهی خصیصه `android:layout_weight` تنظیم نمایید. این مقدار مشخص می کند چه میزان از فضای اضافی موجود به هر یک از المان ها اختصاص داده شود. برای مثال، چنانچه در قالب خود دو `widget` داشته باشید و خصیصه `layout_weight` یکی را با 1 و دیگری را با 2

مقداردهی کنید، در آن صورت اولین المان ۱/۳ فضای موجود را می‌گیرد و دومین المان ۲/۳ فضا را به خود اختصاص می‌دهد. همچنین می‌توانید با قرار دادن مقدار `layout_width` بر روی ۰ یک نسبت مشخص را رعایت کنید.

.ACTIVITY Inflate کردن: تبدیل فایل LAYOUT XML به کلاس view جهت قرار گرفتن در کلاس

## RelativeLayout-1-8-6

در این نوع layout، المان های رابط کاربری (widget) نسب به محل قرارگیری یکدیگر چیده می‌شوند. `relativelayout` را می‌توانید برای ساخت رابط های کاربری پیچیده مورد استفاده قرار دهید. در واقع `layout manager` یک `RelativeLayout` پیچیده و قدرتمند است که فقط باید در شرایط پیچیده مورد استفاده قرار گیرد زیرا این قالب برای چیدمان و تنظیم ظاهر المان های داخل خود به منابع زیاد و محاسبات سنگین احتیاج دارد.

یکی از موارد استفاده `i` `RelativeLayout` زمانی است که می‌خواهید فقط یک کامپوننت را در مرکز صفحه قرار دهید. برای این منظور کافی است یک کامپوننت به `RelativeLayout` اضافه نموده و مقدار خصیصه `i` `true` را بر روی `android:layout_centerInParent` تنظیم کنید.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <ProgressBar
        android:id="@+id/progressBar1"
        style="?android:attr/progressBarStyleLarge"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        />
</RelativeLayout>
```

## GridLayout-1-8-7

برای اولین بار در ویرایش ۴.۰ اندروید ارائه شد. layout نام برده به شما این امکان را می‌دهد تا فضای جاری را خانه بندی کنید و هر view می‌تواند یک یا چند خانه را اشغال کند. در واقع `GridLayout` قالب را مانند یک جدول به خانه، سطر و ستون تقسیم بندی می‌کند.

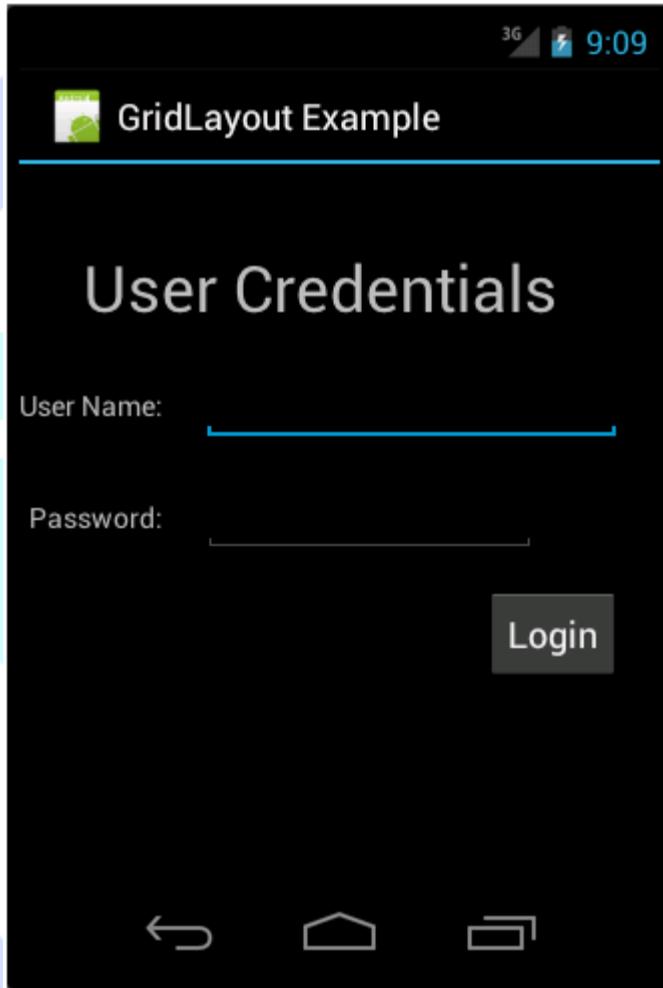
به وسیله‌ی این layout manager شما می‌توانید دقیقاً تعیین کنید هر View چند ستون را به خود تخصیص دهد، در کدام سطر و ستون قرار گیرد و در نهایت چند سطر و ستون را اشغال کند. اگر خودتان به صورت صریح مشخص نکنید، در آن صورت GridLayout مقادیر پیش فرض را تخصیص می‌دهد. برای مثال، به هر view یک ستون و یک سطر اختصاص می‌یابد و محل view نیز به ترتیبی که در کد تعریف کردید مشخص می‌شود.

فایل XML زیر یک قالب با GridLayout تعریف می‌کند.

```
<?xml version="1.0" encoding="utf-8"?>
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/GridLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:columnCount="4"
    android:useDefaultMargins="true" >
    <TextView
        android:layout_column="0"
        android:layout_columnSpan="3"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="40dp"
        android:layout_row="0"
        android:text="User Credentials"
        android:textSize="32dip" />
    <TextView
        android:layout_column="0"
        android:layout_gravity="right"
        android:layout_row="1"
        android:text="User Name: " >
    </TextView>
    <EditText
        android:id="@+id/input1"
        android:layout_column="1"
        android:layout_columnSpan="2"
        android:layout_row="1"
        android:ems="10" />
    <TextView
        android:layout_column="0"
        android:layout_gravity="right"
        android:layout_row="2"
        android:text="Password: " >
    </TextView>
    <EditText
        android:id="@+id/input2"
        android:layout_column="1"
        android:layout_columnSpan="2"
        android:layout_row="2"
        android:inputType="textPassword"
        android:ems="8" />
```

```
<Button
    android:id="@+id/button1"
    android:layout_column="2"
    android:layout_row="3"
    android:text="Login" />
</GridLayout>
```

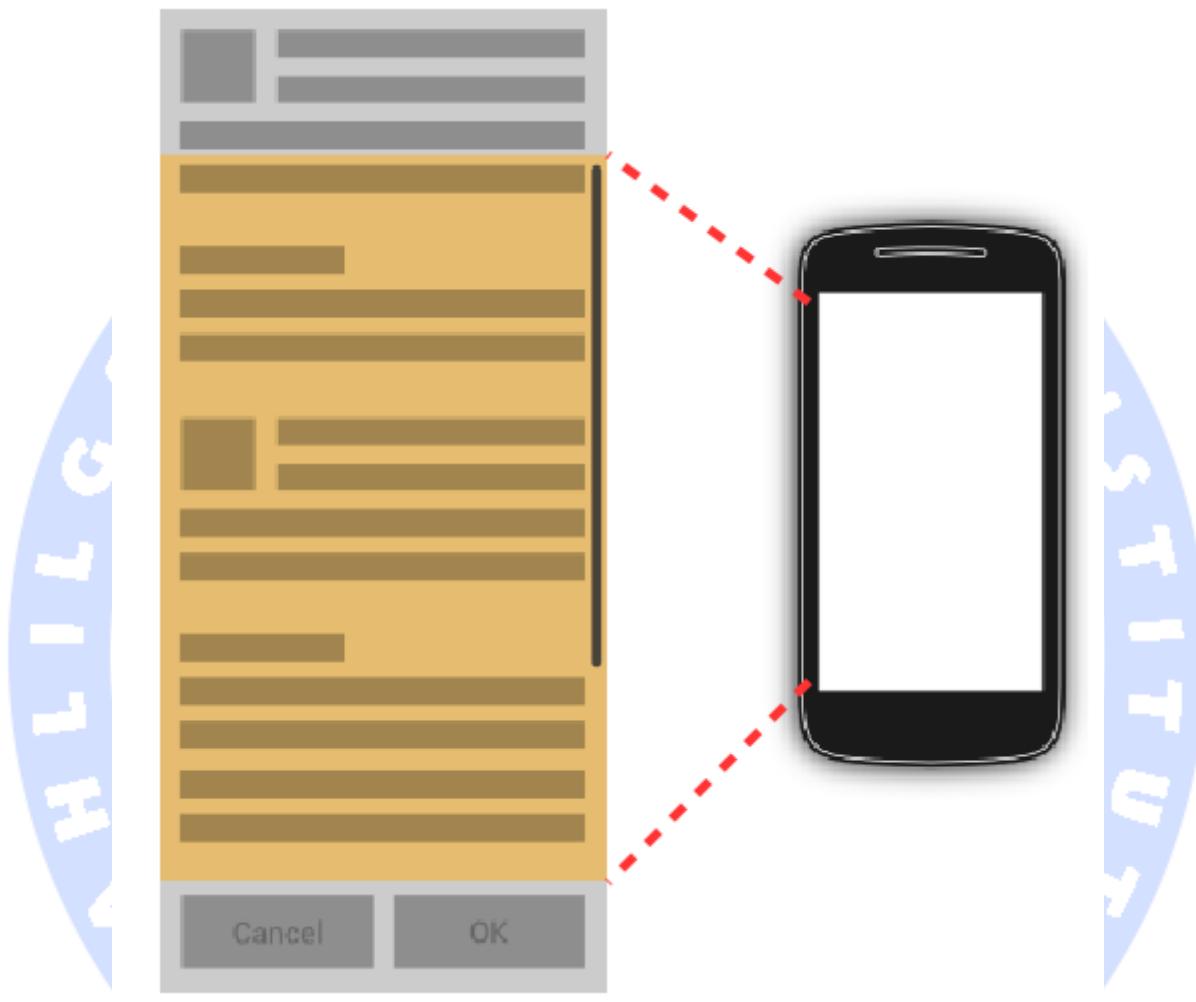
این کد ظاهری مشابه آنچه در تصویر زیر مشاهده می کنید را تعریف می کند.



## ScrollView-1-8-8

گاهی لازم است تمامی view ها را در UI به نمایش بگذارید، حتی اگر فضای کافی برای نمایش آن ها وجود نداشته باشد. در صورتی که view بیش از حد بزرگ باشد، آنگاه ScrollView یک نوار پیمایش به صفحه اضافه نموده و به شما امکان می دهد به وسیله‌ی آن کل محتوا یا المان مورد نظر را (با کشیدن نوار به سمت چپ یا راست) در نمایشگر مشاهده نمایید. لازم به ذکر است که ScrollView یک layout manager هیچ یک HorizontalScrollView محسوب نمی

شوند، بلکه خود می توانند یک view میزبان باشند و آن view تعدادی فرزند در خود داشته باشد.



در زیر یک فایل XML مشاهده می کنید که در آن از ScrollView (جهت نمایش کل محتوای صفحه با نوار پیمایش) استفاده شده است.

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fillViewport="true"
    android:orientation="vertical" >
    <TextView
        android:id="@+id/TextView01"
        android:layout_width="wrap_content"
```

```

    android:layout_height="wrap_content"
    android:paddingLeft="8dip"
    android:paddingRight="8dip"
    android:paddingTop="8dip"
    android:text="This is a header" android:textAppearance="?android:attr/textAppearanceLarge" >
</TextView>
</ScrollView>
```

خصیصه‌ی "android:fillViewport="true" سبب می‌شود که ScrollView کل صفحه را در بر گیرد حتی اگر المان‌ها کوچکتر از کل صفحه باشند و اساساً نیازی به نوار پیمایش وجود نداشته باشد.

## تمرین: ویرایش view layout (تنظیم چیدمان المان‌های رابط

### کاربری) در زمان اجرا

در این تمرین شما تعدادی radio button به قالب (layout) اپلیکیشن که در تمرین قبلی ایجاد کردید، اضافه خواهید نمود. بسته به انتخابی که کاربر می‌کند، چیدمان دکمه از افقی به عمودی و بالعکس تغییر می‌کند.

### ۸-۹-۱- افزودن layout به قالب radio button و radio group

فایل layout را باز نمایید و سپس یک radio group با دو radio button به قالب برنامه‌ی خود اضافه نمایید.

شناسه‌ی المان‌ها را بر اساس جدول زیر مقداردهی کنید.

جدول تخصیص ID به المان‌ها

ID	View
orientation	Radio Group
horizontal	First radio button

### جدول تخصیص ID به المان ها

ID	View
vertical	Second radio button

در حال حاضر کد موجود در فایل XML می باشد. با کمی دقت متوجه می شود که تنها تگ RadioGroup در این layout جدید می باشد.

```
<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"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >
    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/main_input"
        android:layout_alignParentTop="true"
        android:layout_alignParentStart="true"
        android:layout_alignParentEnd="true" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Start"
        android:id="@+id/button"
        android:layout_below="@id/main_input"
        android:layout_alignParentStart="true"
        android:onClick="onClick"/>
    <RadioGroup
        android:id="@+id/orientation"
        android:layout_below="@+id/button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp">
        <RadioButton
            android:id="@+id/horizontal"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Horizontal" >
        </RadioButton>
```

```
<RadioButton  
    android:id="@+id/vertical"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:checked="true"  
    android:text="Vertical" >  
</RadioButton>  
</RadioGroup>  
</RelativeLayout>
```

کد فوق خروجی زیر را در UI به نمایش می گذارد:



## 10-8-1-ویرایش orientation/وضعیت چیدمان radio group در زمان

### اجرا (به صورت dynamic)

تابع `findViewById()` را در کلاس `activity` ویرایش نمایید. سپس با فراخوانی متدهای `findViewById()` کلاس `RadioGroup` را در فایل `layout` (فایل `xml` که ظاهر برنامه را تعریف می‌کند) پیدا کنید.

یک `Listener` داخل `radio group` پیاده سازی نمایید که به تغییرات گوش فرا داده و وضعیت چیدمان یا `orientation` دکمه را بر اساس انتخاب جاری کاربر تغییر دهد. اینکه کدام دکمه انتخاب شده را می‌توان توسط پارامتر `ID` تشخیص داد.

کلاس `RadioGroup` را در قالب متغیر در کلاس `activity` خود اعلام نمایید. سپس متدهای `RadioGroup.OnCheckedChangeListener` را به واسطه این `Listener` اضافه نمایید. این متدهای `setOnCheckedChangeListener()` از `android.widget.RadioGroup` پس از `onCheckedChanged()` این `Listener` (متند) اجرا می‌شوند و متعاقباً دستورات پیاده سازی شده در `onCheckedChanged()` را اجرا می‌کنند.

می‌توانید از قالب آمده ای زیر به عنوان الگو جهت پیاده سازی `Listener` استفاده نمایید.

```
final RadioGroup group1 = (RadioGroup) findViewById(R.id.orientation);
group1.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(RadioGroup group, int checkedId) {
        switch (checkedId) {
            case R.id.horizontal:
                group.setOrientation(LinearLayout.HORIZONTAL);
                break;
            case R.id.vertical:
                group.setOrientation(LinearLayout.VERTICAL);
                break;
        }
    }
});
```

## ۱-۸-۱- تست اپلیکیشن

برنامه‌ی خود را اجرا نموده و انتخاب خود را تغییر دهید. بایستی بر اساس گزینه‌ی مورد انتخاب دکمه‌ها تغییر کند. orientation

## ۱-۹-۱- دسترسی به محتوای static و استفاده از آن

### ۱-۹-۱- دسترسی به فایل‌های محتوا (resources) از طریق کلاس Resources

کلاس Resources به شما اجازه می‌دهد به منابع و محتوا اپلیکیشن خود به صورت مجزا و فردی دسترسی داشته باشید. می‌توانید با فراخوانی متد () getResources() از کلاس Context به راحتی به نمونه‌ی کلاس Resources دسترسی داشته باشید. از آنجایی که کلاس‌های activity و service همگی از (property‌ها و متدهای کلاس) Context ارث بری دارند، شما می‌توانید این متد را مستقیماً در پیاده‌سازی‌های این کامپوننت‌ها بکار ببرید.

سایر کلاس‌های framework اندروید نیز به نمونه‌ای از کلاس Resources احتیاج دارند. برای مثال، کد زیر برای شما نمایش می‌دهد چگونه یک فایل Bitmap از ID اشاره گر ایجاد نمایید.

```
// BitmapFactory requires an instance of the Resource class
BitmapFactory.decodeResource(getResources(), R.drawable.ic_action_search);
```

### ۱-۹-۲- دسترسی به view‌ها از layout در کلاس activity

طبعاً برای تنظیم ظاهر view‌ها، شما لازم دارید که در کد fragment و activity و به آن‌ها دسترسی داشته و property‌های آن‌ها را ویرایش نمایید.

جهت دسترسی به view مورد نظر از layout جاری، کافی است متد findViewByid(id) را فراخوانی نمایید. پارامتر ورودی id در واقع اشاره به خصیصه‌ی ID از view مورد نظر در فایل layout دارد. کد زیر کاربرد این متد را به نمایش می‌گذارد.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    TextView textView = (TextView) findViewById(R.id.mytext);
```

```
// TODO do something with the TextView
}
```

همچنین این امکان وجود دارد که به وسیله‌ی متدهای `findViewById(id)`، در یک `view hierarchy`، مد نظر را پیدا کرده و به آن دسترسی داشت.

همچنین می‌توانید یک متدهای کمکی (utility method) تعریف کنید و به وسیله‌ی آن آسان‌تر به `view` دلخواه دسترسی داشته باشید.

با این روش شما می‌توانید بدون طی کردن پروسه‌ی تبدیل (cast) صریح، آسان‌تر و سریع‌تر به `view` مورد نظر دسترسی داشته باشید.

```
package com.example.android.test;
import android.app.Activity;
import android.view.View;
public class UiUtils {
    public static <T extends View> T findViewById(View root, int id) {
        return (T) root.findViewById(id);
    }
    public static <T extends View> T findViewById(Activity activity, int id) {
        return (T) activity.getWindow().getDecorView().getRootView().findViewById(id);
    }
    // search in the layout of the activity
    LinearLayout linearLayout = (LinearLayout) findViewById(R.id.mylayout);
    // afterwards search in linearLayout for another view
    TextView textView = (TextView) linearLayout.findViewById(R.id.mytext);
    // note, you could have directly searched for R.id.mytext, the above coding
    // is just for demonstration purposes
}
```

```
package com.example.android.test;
import android.app.Activity;
import android.view.View;
public class UiUtils {
    public static <T extends View> T findViewById(View root, int id) {
        return (T) root.findViewById(id);
    }
    public static <T extends View> T findViewById(Activity activity, int id) {
        return (T) activity.getWindow().getDecorView().getRootView().findViewById(id);
    }
}
```

Button button = UiUtils.findViewById(this, R.id.button);

### 1-9-3- دسترسی به محتوای مورد نظر در فایل های XML از دیگر فایل های

resource

در فایل های XML خود، برای مثال فایل های layout اپلیکیشن، می توانید به سایر resource ها اشاره کرده و در واقع از داخل فایل مورد نظر به آن ها دسترسی داشته باشید. این کار از طریق درج علامت @ صورت می پذیرد.

برای مثال، جهت دسترسی به مقدار color که در یک فایل XML (resource file) تعریف شده، کافی است این ساختار نگارشی را بکار ببرید: @color/your\_id. همچنین ممکن است قبل از یک String با ID یا شناسه‌ی "titlepage" در فایل XML تعریف کرده باشید و اکنون می خواهید به آن دسترسی داشته باشید. می توانید با دستور @string/titlepage به راحتی به آن متغیر دسترسی داشته باشید.

به منظور دسترسی به محتوایی که خود سیستم اندروید ارائه می دهد (system resource)، لازم است namespace یا پوشه‌ی android را در اشاره گرها (id هایی که برای دسترسی به آن منبع مورد استفاده قرار می دهید) لاحظ نمایید (مثال: android.R.string.cancel).

## ۱-۹-۴-استفاده از پوشه `assets` و دسترسی به داده های ذخیره شده در آن

در سیستم اندروید، پوشه ای به نام `res` تعریف شده که دربردارندهٔ مقادیر ساخت یافته و سازماندهی شده (با `semantics` و معانی از پیش تعریف شده) ویژهٔ محیط کاری اندروید است. هر نوع داده ای را می‌توان در این پوشه ذخیره کرد. شما می‌توانید به فایل‌های ذخیره شده در پوشه، بر اساس محل قرارگیری آن فایل دسترسی داشته باشید. داخل این پوشه `assets` همچنین می‌توانید زیرپوشه داشته باشید. برای نگهداری داده‌هایی که با ساختار مشخصی سازمان‌دهی و ذخیره نشده‌اند (`unstructured data`)، می‌توانید از پوشه `res/raw` استفاده نمایید. هر چند بهتر است این نوع اطلاعات را نیز در همان پوشه `assets` ذخیره نمایید (منابعی که در پوشه `res` ذخیره می‌شود برای کاربر قابل دسترسی هستند).

می‌توانید با فراخوانی متدهای `getAssets()` که نمونه ای از کلاس `AssetManager` را برمی‌گرداند، به فایل‌های خام ذخیره شده در پوشه `res/raw` دسترسی داشته باشید. این متدهای کلاس `Context` برگرفته شده است.

در زیر یک آبجکت از کلاس `AssetsManager` ایجاد می‌کنید که این کلاس اجازهٔ دسترسی به فایل‌های خام مستقر در پوشه `assets` را می‌دهد. اسم این آبجکت را `manager` انتخاب می‌کنید. سپس با فراخوانی متدهای `getAssets()` نمونه ای از کلاس `AssetsManager` را برمی‌گردانید. از کلاس `InputStream` یک آبجکت به نام `open` ایجاد می‌کنید. سپس متدهای `open()` را بر روی آبجکت ایجاد شده از کلاس `InputStream` فراخوانی می‌کنید و اسم فایل مورد نظر را به عنوان پارامتر ورودی به متدهای `open()` پاس می‌دهید. این متدهای `open()` در آبجکت `open()` از کلاس `InputStream` می‌ریزد.

```
// get the AssetManager
AssetManager manager = getAssets();
// read the "logo.png" bitmap from the assets folder
InputStream open = null;
try {
    open = manager.open("logo.png");
    Bitmap bitmap = BitmapFactory.decodeStream(open);
    // assign the bitmap to an ImageView in this layout
```

```

ImageView view = (ImageView) findViewById(R.id.imageView1);
view.setImageBitmap(bitmap);
} catch (IOException e) {
    e.printStackTrace();
} finally {
    if (open != null) {
        try {
            open.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
}

```

## تمرین: استفاده از محتوا/resource های XML و در کد



### ۹-۱-۱- افزودن عکس به اپلیکیشن

در این تمرین نیز اپلیکیشن قبلی را ادامه می دهید. دو تصویر جدید ایجاد می کنید و آن را به پروژه ای به نام ic\_tethering و ic\_offline اضافه می نمایید.



### ۹-۱-۲- افزودن view جدید به پروژه

فایل layout خود را باز نموده و یک تگ Button و ImageView به آن اضافه نمایید. سپس فایل layout را از طریق فایل ImageView به ImageView اضافه کنید.

```

<!--
    NOTE: More attributes are required
    for the correct layout of the ImageView. These are left
    out for brevity
-->
<ImageView
    android:id="@+id/myicon"
    .... more attributes
    android:src="@drawable/ic_offline" />

```

### ۹-۱-۳- جایگزین کردن عکس ها با کلیک بر روی دکمه

زمانی که بر روی دکمه کلیک می شود، با استفاده از متد `findViewByIId()` به `ImageView` اشاره دسترسی پیدا می کنید. سپس با استفاده از متد `SetImageResource()` فایل png را (که در زمان اجرای برنامه به صورت یک آبجکت `Drawable` نمایش داده می شود) به `ImageView` خود اضافه نمایید. پارامتر ارسال شده به متد `SetImageResource()` یک شناسه یا ID است که به فایل R.drawable.your\_png\_file اشاره دارد و شناسه ی متناظر فایل مورد نظر می باشد.

## 1-9-8- تست اپلیکیشن

برنامه را اجرا نمایید. حال بر روی دکمه کلیک کنید. در پی کلیک بر روی دکمه ی جدید، تصویری که در UI قابل مشاهده می باشد باید با تصویر دیگری جایگزین شود.

## تمرین: استفاده از ScrollView در پروژه

در این تمرین یک کلاس ScrollView به پروژه اضافه می کنید که یک نوار پیمایش به اپلیکیشن اضافه نموده و امکان مشاهده ی تمامی بخش های view را در صفحه ی نمایشگر برای کاربر فراهم می آورد. یک پروژه اندروید به نام de.vogella.android.scrollView ایجاد کرده و سپس کلاس activity با نام ScrollViewActivity را به آن اضافه نمایید.

activity را به عنوان فایلی که اپلیکیشن ظاهر برنامه را بر اساس آن طراحی می کند (layout)، انتخاب نمایید.

فایل layout مذبور را که در activity خود فراخوانی می کنید، به صورت زیر ویرایش نمایید:

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fillViewport="true"
    android:orientation="vertical" >
    <LinearLayout
        android:id="@+id/LinearLayout01"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical" >
        <TextView
            android:id="@+id/TextView01"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:paddingLeft="8dip"
            android:paddingRight="8dip"
            android:paddingTop="8dip"
            android:text="This is a header"
            android:textAppearance="?android:attr/textAppearanceLarge" >
        </TextView>
        <TextView
            android:id="@+id/TextView02"
            android:layout_width="wrap_content"
```

```

        android:layout_height="match_parent"
        android:layout_weight="1.0"
        android:text="@+id/TextView02" >
    </TextView>
    <LinearLayout
        android:id="@+id/LinearLayout02"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" >
        <Button
            android:id="@+id/Button01"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1.0"
            android:text="Submit" >
        </Button>
        <Button
            android:id="@+id/Button02"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1.0"
            android:text="Cancel" >
        </Button>
    </LinearLayout>
</LinearLayout>
</ScrollView>

```

کد کلاس ScrollViewActivity را به صورت زیر ویرایش نمایید.

```

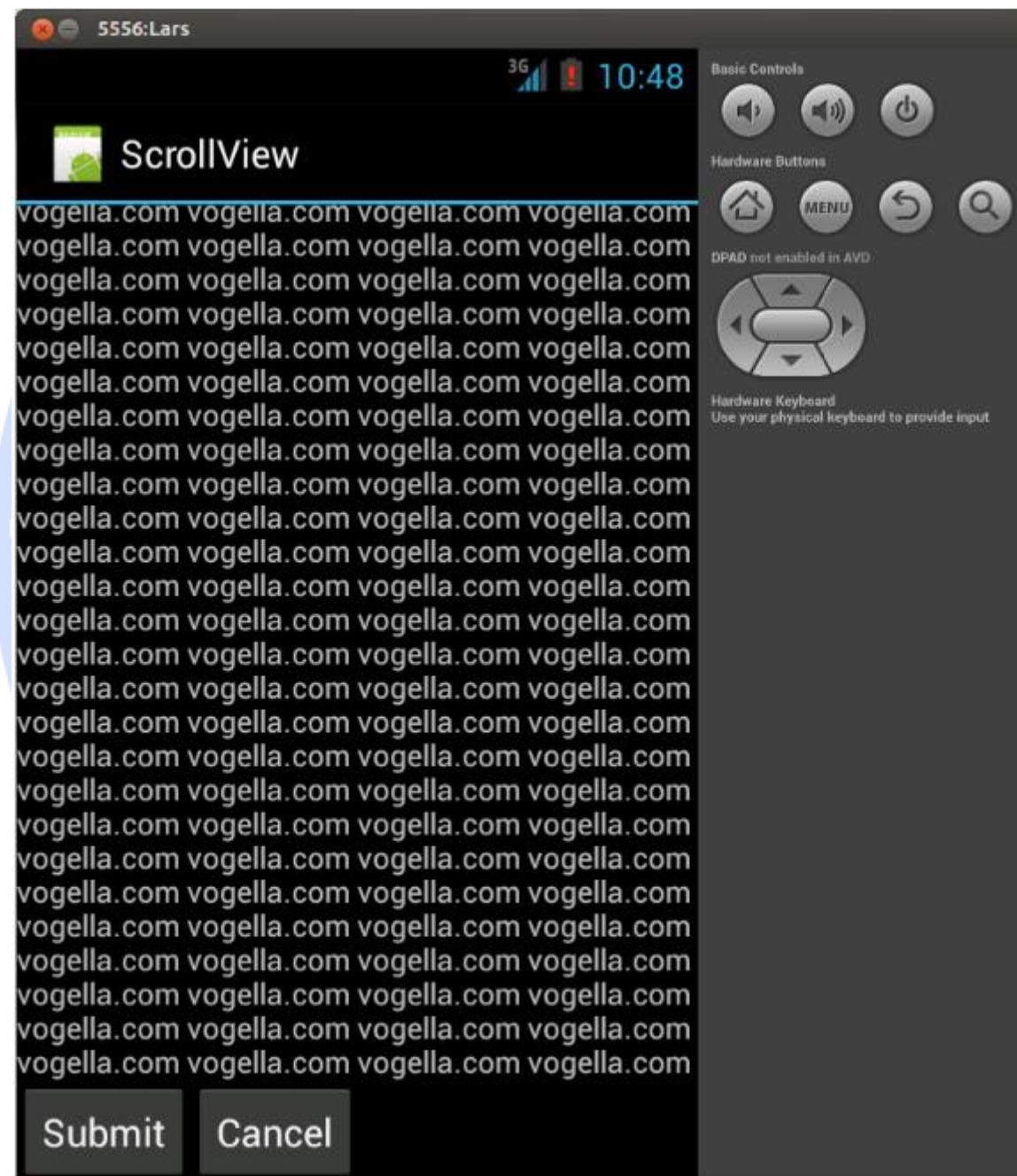
package de.vogella.android.scrollView;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;
public class ScrollViewActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        TextView view = (TextView) findViewById(R.id.TextView02);
        String s="";
        for (int i=0; i < 500; i++) {
            s += "vogella.com ";
        }
        view.setText(s);
    }
}

```

اپلیکیشن خود را اجرا کنید. با توجه به کدی که برای برنامه نوشته اید، بایستی بتوانید با استفاده از نوار پیمایش به پایین صفحه دسترسی داشته و دکمه ها را مشاهده نمایید.

9-1-تست اپلیکیشن

اکنون می بایست با کلیک بر روی دکمه **جدید**، تصویر قابل مشاهده در نمایشگر را جایگزین نماید.



### تمرین: نوشتن برنامه‌ی تبدیل دما

در طی این تمرین شما نحوه‌ی ایجاد و استفاده از `resource` و محتوای اندروید را خواهید آموخت.

این اپلیکیشن از قبل نوشته شده و به صورت آماده در آدرس <http://play.google.com/store/apps/details?id=de.vogella.android.temperature> از طریق Google Play قابل دسترسی می‌باشد.

می‌توانید با گوشی اندروید خود بارکد زیر را اسکن کرده، سپس برنامه‌ی تبدیل دما را از طریق اپلیکیشن Google Play دانلود و نصب نمایید.



### 1-9-10-ساخت پروژه

یک پروژه‌ی جدید اندروید بر اساس پارامترهای ارائه شده در جدول زیر ایجاد نمایید.

Table 7. New Android project

Property	Value
Application Name	Temperature Converter
Package name	com.vogella.android.temperatureconverter
API (Minimum, Target, Compile with)	Latest

Table 7. New Android project	
Property	Value
Template	Empty Activity
Activity	MainActivity
Layout	activity_main

### ۱-۹-۱- ایجاد attribute ها

جهت ویرایش فایل res/values/strings.xml آن را انتخاب و سپس بر روی آن دابل کلیک نمایید.  
دو متغیر String و Color بر اساس جدول زیر اضافه نمایید.

Table 8. New attributes to add		
Type	Name	Value
Color	myColor	#F5F5F5
String	celsius	to Celsius
String	fahrenheit	to Fahrenheit
String	calc	Calculate

مقادیر می بایست به صورت زیر در فایل XML درج شود.

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<resources>
<string name="app_name">Temperature Converter</string>
<string name="action_settings">Settings</string>
<string name="hello_world">Hello world!</string>
<color name="myColor">#F5F5F5</color>
<string name="102ahrenh">to Celsius</string>
<string name="102ahrenheit">to Fahrenheit</string>
<string name="calc">Calculate</string>
</resources>
```

## 12-9-1- ایجاد فایل Layout و تنظیم کننده‌ی ظاهر اپلیکیشن

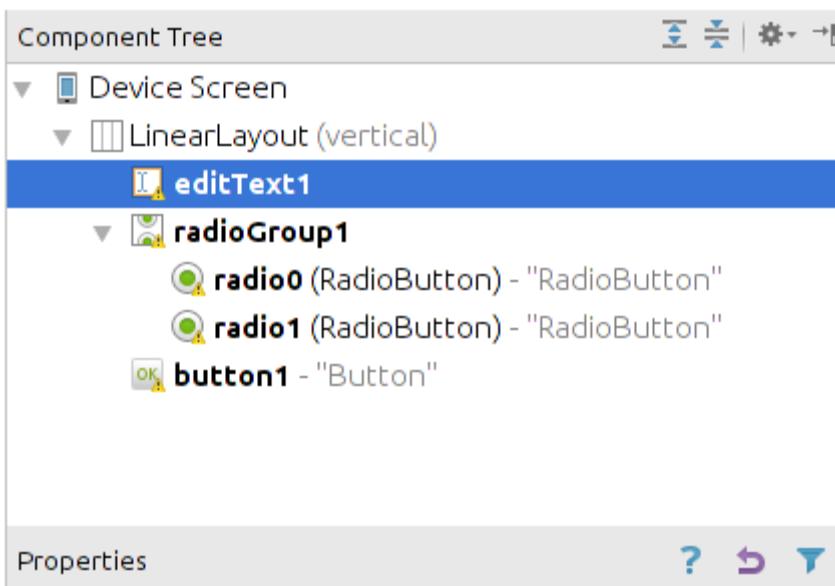
فایل XML را انتخاب نموده و با دوبار کلیک بر روی آن ویرایشگر متناظر را باز نمایید.

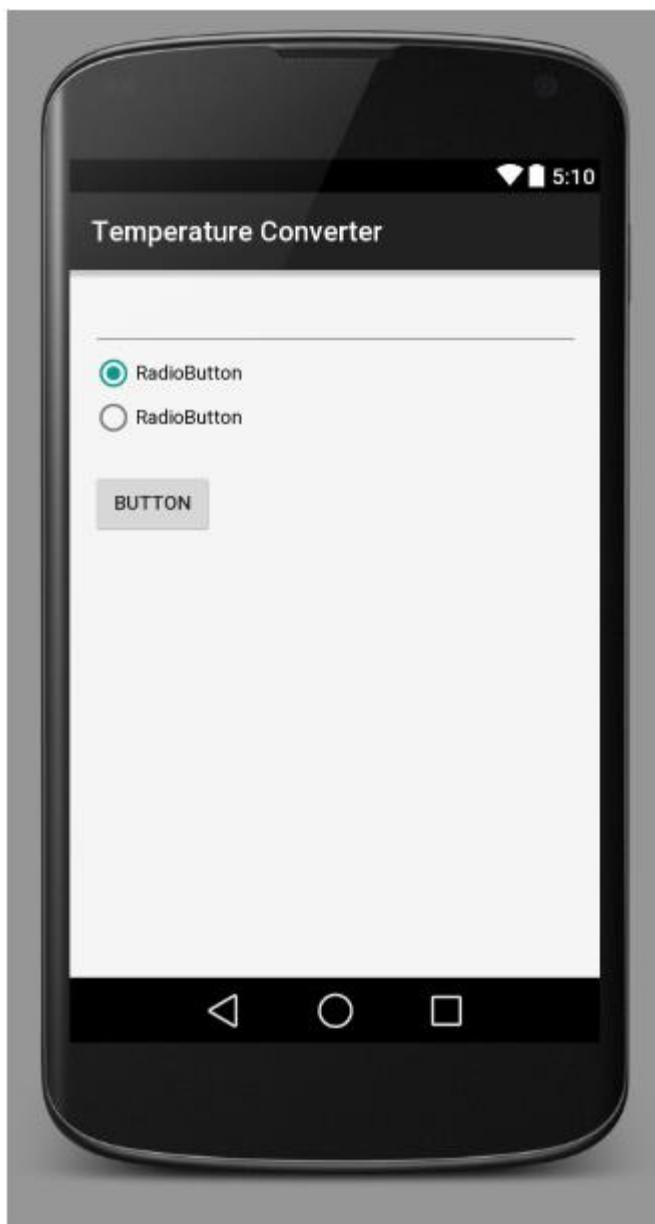
تمامی view‌ها را از layout حذف نمایید. این کار را می‌توانید یا به طور مستقیم از طریق فایل XML یا به وسیله‌ی ویرایشگر گرافیکی انجام دهید.

سپس یک LinearLayout اضافه نموده و بعد از آن یک ViewText دیگر با EditText به عنوان فرزند در داخل آن تعریف کنید.

پس از آن یک RadioGroup به همراه دو radio button به layout خود اضافه نمایید. این کار را می‌توانید یا از طریق فایل XML یا ویرایشگر گرافیکی انجام دهید. یک روش ساده برای سازمان دهی کامپوننت‌ها این است که آن‌ها را با اشاره گر موس کشیده و بر روی Component Tree view جایگذاری نمایید.

خروجی می‌بایست به صورت زیر باشد.





اکنون بر روی تب XML فایل layout خود کلیک نمایید. آن را با کد زیر تطبیق داده و از مشابه بودن آن اطمینان حاصل نمایید.

**توجه:** تیم توسعه دهندگان ابزار اندروید این کد را هر چند وقت یکبار تغییر می دهند، به همین خاطر اگر فایل XML شما کمی متفاوت بود، جای تعجب نیست.

```
<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"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity"
    android:background="@color/myColor">
<EditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/editText1" />
<RadioGroup
    android:id="@+id/radioGroup1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignStart="@+id/editText1"
    android:layout_below="@+id/editText1">
    <RadioButton
        android:id="@+id/radio0"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:checked="true"
        android:text="RadioButton" />
    <RadioButton
        android:id="@+id/radio1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="RadioButton" />
</RadioGroup>
<Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignStart="@+id/radioGroup1"
    android:layout_below="@+id/radioGroup1"
    android:layout_marginTop="22dp"
    android:text="Button" />
</LinearLayout>
```

**توجه:** با تعدادی پیغام هشدار مواجه می شوید. این خطاها را در بخش بعدی تمرین برطرف خواهید نمود.

You see some warning messages. You fix these in the following section of this exercise.

## 1-9-13-ویرایش مقادیر های view property

نسخه ی XML فایل را باز نمایید و مقدار property را به @string/Celsius را به خصوصیت radio button از اولین تخصیص دهید. سپس رشته ی farenheit را به خصوصیت android:text از دومین radio button انتساب دهید.

```

<RadioGroup
    android:id="@+id/radioGroup1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/editText1"
    android:layout_below="@+id/editText1" >

    <RadioButton
        android:id="@+id/radio0"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:checked="true"
        android:text="@string/celsius" />

    <RadioButton
        android:id="@+id/radiol"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/fahrenheit" />
</RadioGroup>

```

مقدار خصوصیت Checked را بر روی true تنظیم نمایید و سپس مقدار onClick را به خصوصیت اختصاص دهید.

مقدار @string/calc را به خصوصیت text اولین دکمه اختصاص داده، سپس مقدار onClick را به خصوصیت onClick انتساب دهید.

خصوصیت numberDecimal و numberSigned با EditText مقداردهی نمایید. این اصلاحات در فایل XML زیر قابل مشاهده می باشد. در پایان ID آن را به "InputValue" تغییر دهید.

```

<EditText
    android:id="@+id/inputValue"
    android:layout_width="match_parent"

```

```
    android:layout_height="wrap_content"
    android:layout_alignParentEnd="true"
    android:layout_below="@+id/textView"
    android:ems="10"
    android:inputType="numberSigned|numberDecimal" />
```

تمامی کامپونت های UI در یک layout با فرمت XML تعریف می شوند. رنگ پس زمینه را نیز در این فایل اضافه نمایید.

Color را انتخاب نموده و سپس myColor را در کادر محاوره ای انتخاب نمایید. رنگ پس زمینه UI در فایل XML به صورت زیر در آخرین خط تنظیم می شود.

```
<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"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity"
    android:background="@color/myColor">
```

پس از افزودن این خط کد، پس زمینه بایستی با رنگ whitesmoke تنظیم شود. هر چند تشخیص این تغییر ممکن است با توجه به رنگ انتخابی کمی دشوار باشد.

حال بر روی تب activity\_main.xml کلیک نموده و با بررسی آن از صحیح بودن کامل کد اطمینان حاصل نمایید.

```
<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"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity"
    android:background="@color/myColor">
    <EditText
        android:layout_width="match_parent"
```

```

        android:layout_height="wrap_content"
        android:id="@+id/inputValue"
        android:inputType="numberSigned|numberDecimal"/>
<RadioGroup
    android:id="@+id/radioGroup1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignStart="@+id/editText1"
    android:layout_below="@+id/editText1">
    <RadioButton
        android:id="@+id/radio0"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:checked="true"
        android:text="@string/celsius" />
    <RadioButton
        android:id="@+id/radio1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/fahrenheit" />
</RadioGroup>
<Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignStart="@+id/radioGroup1"
    android:layout_below="@+id/radioGroup1"
    android:layout_marginTop="22dp"
    android:text="@string/calc"
    android:onClick="onClick"/>
</LinearLayout>

```

### 14-1- تعریف یک کلاس کمکی (utility class)

یک کلاس تعریف کرده و در آن کد لازم برای انجام عملیات تبدیل از واحد سلسیوس به فارانهایت و بالعکس را بنویسید.

```

package com.vogella.android.temperatureconverter;
public class ConverterUtil {
    // converts to celsius
    public static float convertFahrenheitToCelsius(float fahrenheit) {
        return ((fahrenheit - 32) * 5 / 9);
    }
    // converts to fahrenheit
    public static float convertCelsiusToFahrenheit(float celsius) {
        return ((celsius * 9) / 5) + 32;
    }
}

```

## activity-1-9-15 ویرایش کد

کد موجود در کلاس MainActivity را به صورت زیر ویرایش کنید:

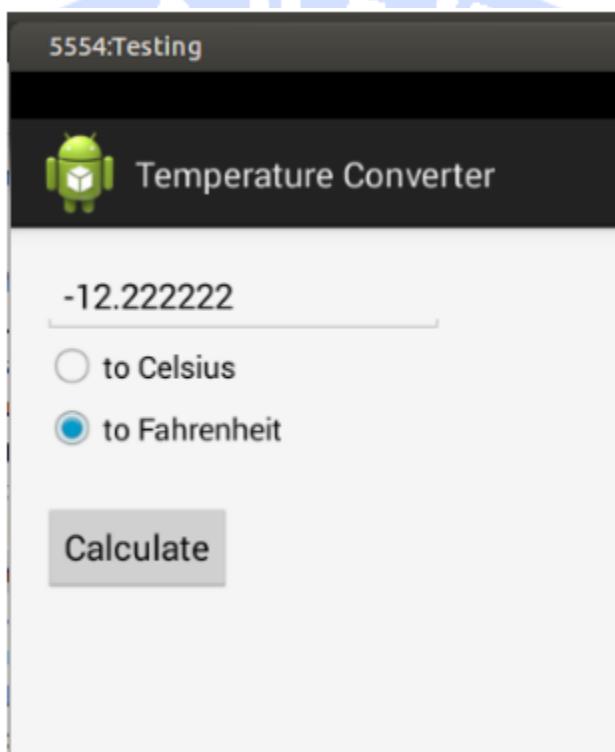
```

package com.vogella.android.temperatureconverter;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.Toast;
public class MainActivity extends Activity {
    private EditText text;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        text = (EditText) findViewById(R.id.inputValue);
    }
    // this method is called at button click because we assigned the name to the
    // "OnClick" property of the button
    public void onClick(View view) {
        switch (view.getId()) {
            case R.id.button1:
                RadioButton celsiusButton = (RadioButton) findViewById(R.id.radio0);
                RadioButton fahrenheitButton = (RadioButton) findViewById(R.id.radio1);
                if (text.getText().length() == 0) {
                    Toast.makeText(this, "Please enter a valid number",
                        Toast.LENGTH_LONG).show();
                    return;
                }
                float inputValue = Float.parseFloat(text.getText().toString());
                if (celsiusButton.isChecked()) {
                    text.setText(String.valueOf(ConverterUtil.convertFahrenheitToCelsius(inputValue)));
                    celsiusButton.setChecked(false);
                    fahrenheitButton.setChecked(true);
                } else {
                    text.setText(String.valueOf(ConverterUtil.convertCelsiusToFahrenheit(inputValue)));
                    fahrenheitButton.setChecked(false);
                    celsiusButton.setChecked(true);
                }
                break;
        }
    }
}

```

## 9-1- راه اندازی اپلیکیشن

اپلیکیشن اندروید خود را راه اندازی نموده و یک عدد جهت تبدیل در آن وارد نمایید. سپس تبدیل را انتخاب نموده و دکمه‌ی مربوطه را کلیک نمایید. پس از مشاهده‌ی نتیجه گزینه‌ی اول، جهت اطمینان حاصل نمودن از عملکرد صحیح برنامه در تبدیل بالعکس، گزینه‌ی دیگر را انتخاب نمایید.



## 10- فرایند آماده سازی، عرضه، توزیع و نصب اپلیکیشن (Deployment)

### 10-1- نحوه‌ی آماده سازی و عرضه‌ی اپلیکیشن برای نصب

روش‌های مختلفی برای نصب اپلیکیشن بر روی دستگاه وجود دارد. شما می‌توانید اپلیکیشن را از طریق USB و در حالت تست بر روی دستگاه مستقر نمایید یا آن را از طریق ایمیل به خود ارسال نموده و نصب کنید و یا اپلیکیشن را بر روی فروشگاه‌های مجازی عرضه‌ی اپلیکیشن‌های تحت موبایل اندروید بارگذاری نموده و از آنجا امکان دانلود و نصب آن را فراهم نمایید. در زیر روش‌های معمول و پرطرفدار عرضه و نصب اپلیکیشن تشریح شده‌اند.

## 2-10-1-مشخص کردن اجزا نرم افزاری و سیستم سخت افزاری مورد نیاز

### برای اپلیکیشن

شما می توانید با تعریف تگ <uses-feature> در فایل تنظیمات اندروید (manifest) مشخص نمایید اپلیکیشن شما برای اجرای موفقیت آمیز بر روی دستگاه به چه ابزاری نیاز دارد. با تنظیم مقدار خصیصه‌ی (attribute) android:required در روی true شما به دستگاه اعلان می کنید که برای اجرای اپلیکیشن بایستی از سخت افزار و نرم افزار مربوطه برخوردار باشد یا چنانچه آن ویژگی در دستگاه تعییه شده باشد برنامه بهتر کار می کند اما در عین حال به گونه ای طراحی شده که بتواند بدون حضور آن ویژگی نیز با موفقیت اجرا شود. برای مثال، اپلیکیشنی را در نظر بگیرید که برای عملکرد خود به یک حسگر یا دوربین احتیاج دارد. کافی است تگ نام برد و را در فایل تنظیمات این برنامه تعریف کرده و مقدار خصیصه‌ی android:required آن را بر روی true تنظیم نمایید.

## 2-10-3 Sign کردن (تخصیص امضای الکترونیکی جهت دیباگ) اپلیکیشن

### قبل از انتشار و عرضه‌ی آن در فروشگاه‌های مجازی

اپلیکیشن‌های اندروید بایستی قبل از اینکه بر روی دستگاه نصب شوند، امضا شده و کلید یا امضای الکترونیکی منحصر بفردی به آن اختصاص یابد. به هنگام ساخت و توسعه‌ی اپلیکیشن، پروسه‌ی کامپایل و build برنامه‌ی شما را با یک debug key (امضا الکترونیکی جهت دیباگ) امضا می کند.

جهت نصب اپلیکیشن اندروید، شما می بایست فایل apk اندروید را با یک کلید منحصر بفرد (signature key) که به صورت خودکار تولید می شود، امضا نمایید.

لازم به ذکر است که برای بروز رسانی اپلیکیشن خود، می بایست از همان signature key در فروشگاه مجازی Google Play استفاده نمایید. اگر این کلید منحصر بفرد را از دست بدید، دیگر هیچ وقت نمی توانید برنامه‌ی خود را بروز رسانی کنید. توصیه می شود از این کلید نسخه‌ی پشتیبان تهیه نمایید.

## 10-1- خروجی امضا شده دادن (Export) اپلیکیشن از طریق محیط کاری

### Android Studio

جهت خروجی دادن اپلیکیشن از داخل محیط Android Studio منوی اصلی را باز نموده و سپس .Build ▶ Generate Signed APK مسیر روبرو را طی نمایید:

## 10-1- خروجی امضا شده دادن/ Export اپلیکیشن از طریق محیط برنامه

### Eclipse (IDE)

جهت خروجی دادن اپلیکیشن از محیط کاری Eclipse، کافی است بر روی آن راست کلیک کرده و .Android Tools ▶ Export Signed Application Package سپس این مسیر را طی نمایید:

ویزارد به شما اجازه می دهد یک کلید جدید ایجاد کنید یا همان کلید از پیش آماده را مورد استفاده قرار دهید.

## 10-1- نصب اپلیکیشن از روش های دیگر

اندروید به شما اجازه می دهد تا اپلیکیشن را به طور مستقیم نصب نمایید. کافی است بر روی لینکی که در یک پیوست ایمیل یا صفحه ی وب به فایل apk اشاره دارد، کلیک کنید. اندروید از شما می پرسد آیا مایلید اپلیکیشن را نصب نمایید یا خیر.

البته برای این منظور می بایست به بخش تنظیمات دستگاه مراجعه نموده و اجازه ی نصب اپلیکیشن هایی که خارج از فروشگاه های مجازی معتبر عرضه می شود را به طور صریح اعطا نمایید. می توانید به گزینه ی مورد نظر در بخش Security تنظیمات دستگاه خود دسترسی داشته باشید.

## 10-1- فروشگاه مجازی Google Play

جهت عرضه ی اپلیکیشن های اندرویدی خود در Google play می بایست مبلغ 25 دلار را برای یکبار پرداخت کنید. پس از پرداخت این مبلغ می توانید برنامه های خود را به همراه آیکون های Google Play تحت آدرس <https://play.google.com/apps/publish> مسقیما در بارگذاری نمایید.

گوگل ابتدا برنامه را اسکن کرده و از عدم وجود بدافزار در آن اطمینان حاصل می کند. معمولاً چند دقیقه پس از بارگذاری، اپلیکیشن برای دانلود و نصب در دسترس قرار می گیرد.



### نوار ابزار محیط برنامه نویسی 11-1

نوار ابزار محیط کاری Android Studio دربردارنده‌ی تعداد زیادی گزینه ویژه‌ی ساخت و توسعه‌ی اپلیکیشن‌های اندرویدی می باشد. در زیر به هر یک از ابزار موجود در این ToolBar اشاره شده است.



### ابزار های موجود در toolbar محیط برنامه نویسی اندروید استودیو

Number	Action
1	AVD Manager
2	Sync Project with Gradle Files
3	Project Structure / Settings
4	Android SDK manager

### 11-2- محل ذخیره سازی پروژه های اندروید

تمامی پروژه های شما را به صورت پیش فرض در پوشه `i home` کاربر تحت نام `AndroidStudioProjects` ذخیره می کند.

پوشه `i main` دربردارنده فایل های تنظیمات محیط کاری و فایل های `Gradle build` می باشد (عملیات پیکربندی پروژه در فایل `gradle` صورت می پذیرد). در این فایل شما می توانید متدها و خواصی را تعریف کرده و وظایف معینی را مشخص نمایید. فایل های مربوط به خود اپلیکیشن در پوشه `i app` جایگذاری می شود. این پوشه فایل های جاوایی را شامل شده و پیش فرض دربردارنده یک فایل به نام `MainActivity.java` می باشد که برنامه به هنگام راه اندازی از آن شروع می شود. این ساختار در زیر به نمایش گذاشته شده است.

**توجه:** Android Studio این فایل ها را با نمای خاصی به نمایش می گذارد. برای مثال فایل های حاوی تعاریف رشته ها، رنگ ها و محتویات (resource) اپلیکیشن را تحت گره (node) `res` سازمان دهی می کند که با ساختار واقعی فایل یکسان نیست.



### 11-1- باز کردن پروژه ها و راه گزینی (سویچ) بین آن ها

محیط برنامه نویسی Android Studio هر پروژه را در پنجره‌ی متفاوتی به نمایش می‌گذارد. جهت باز کردن پروژه‌ی جاری کافی است مسیر... ▶ File ▶ Open... را طی نمایید. به دنبال آن، پنجره‌ی جدیدی پدیدار شده و پروژه‌ی مورد نظر را در خود نمایش می‌دهد.

جهت بستن پروژه نیز کافی است این مسیر را طی نمایید: File ▶ Close Project. سپس یک پروژه ی جدید ایجاد نموده یا پروژه ی از قبل موجود را به روش یاد شده در بالا باز نمایید.

## 11-1-پاک کردن محتواهای پوشه های build و کامپایل مجدد پروژه/هماهنگ سازی پروژه با Gradle

متاسفانه ابزار build در محیط کاری Android Studio تغییرات وارد شده به فایل های محتوا (res) را نمی شناسد و خود را با آن تطبیق نمی دهد. اگر می خواهید پروژه را کاملاً rebuild نمایید، کافی است مسیر Build ▶ Clean Project را طی کنید.

تغییرات را در فایل gradle build باید به صورت دستی وارد نمایید. جهت هماهنگ سازی پروژه با Gradle، لازم است این مسیر را طی نمایید: Tools ▶ Android ▶ Sync Project with Gradle و بر روی لینک مربوطه در tooltip Graddle build فایل در اختیار شما قرار می دهد، کلیک نمایید.

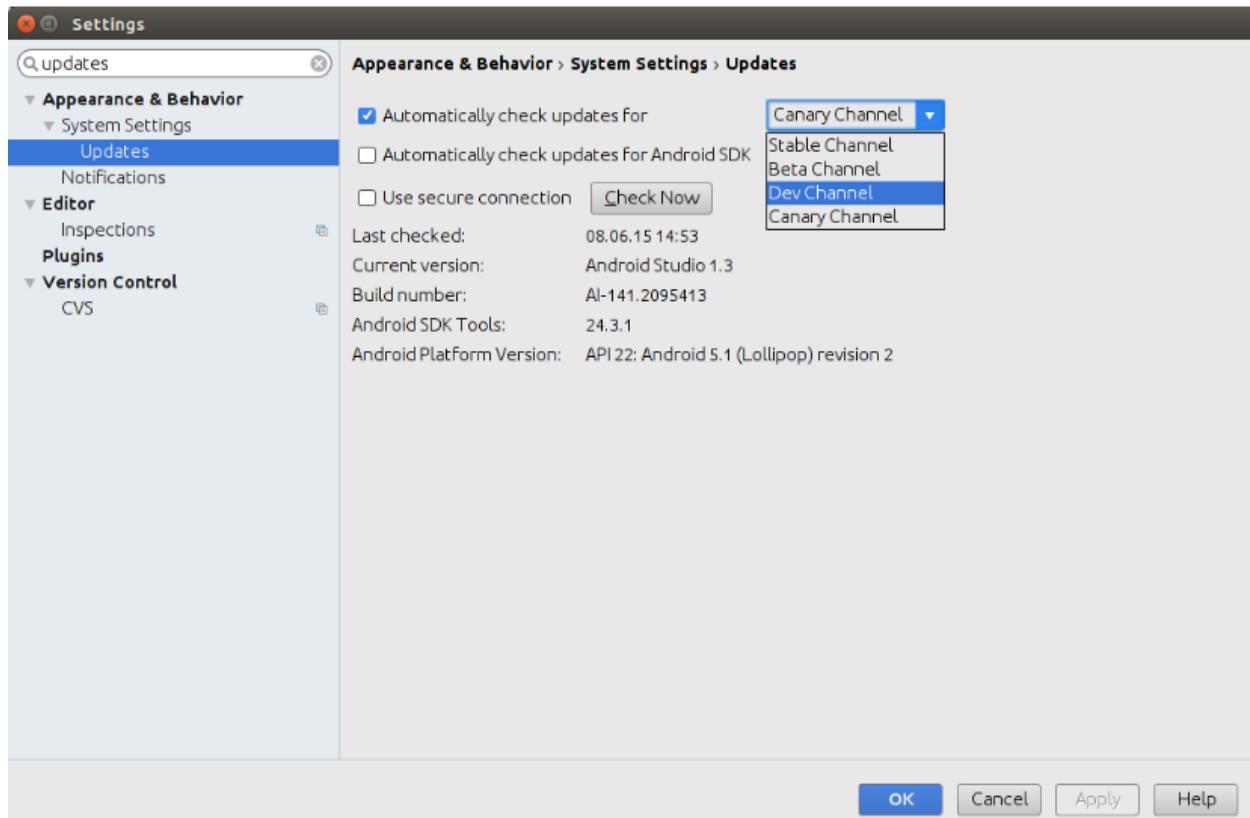
## 11-1-بروز رسانی تنظیمات Android Studio

در نسخه ها (flavor) مختلف ارائه می شود. برنامه نویس می تواند نسخه ی ثابت و نهایی را مورد استفاده قرار دهد یا ویژگی های نسخه های جدید ولی غیرنهایی و در حال تغییر را تست کند.

1. Canary channel :Canary build های build، در واقع نسخه هایی با جدیدترین ویژگی های اضافه شده به محیط کاری هستند و به صورت هفتگی منتشر می شوند. اگرچه این نسخه ها تست شده هستند، اما همچنان احتمال بروز خطا در آن ها وجود دارد چرا که با عجله و با هدف عرضه یا معرفی امکانات جدید انتشار می یابند. از این جهت استفاده از آن برای production (زمان و مرحله ای که کاربر محیط توسعه را بر روی سیستم نصب می کند و در آن به برنامه سازی می پردازد) توصیه نمی شود.

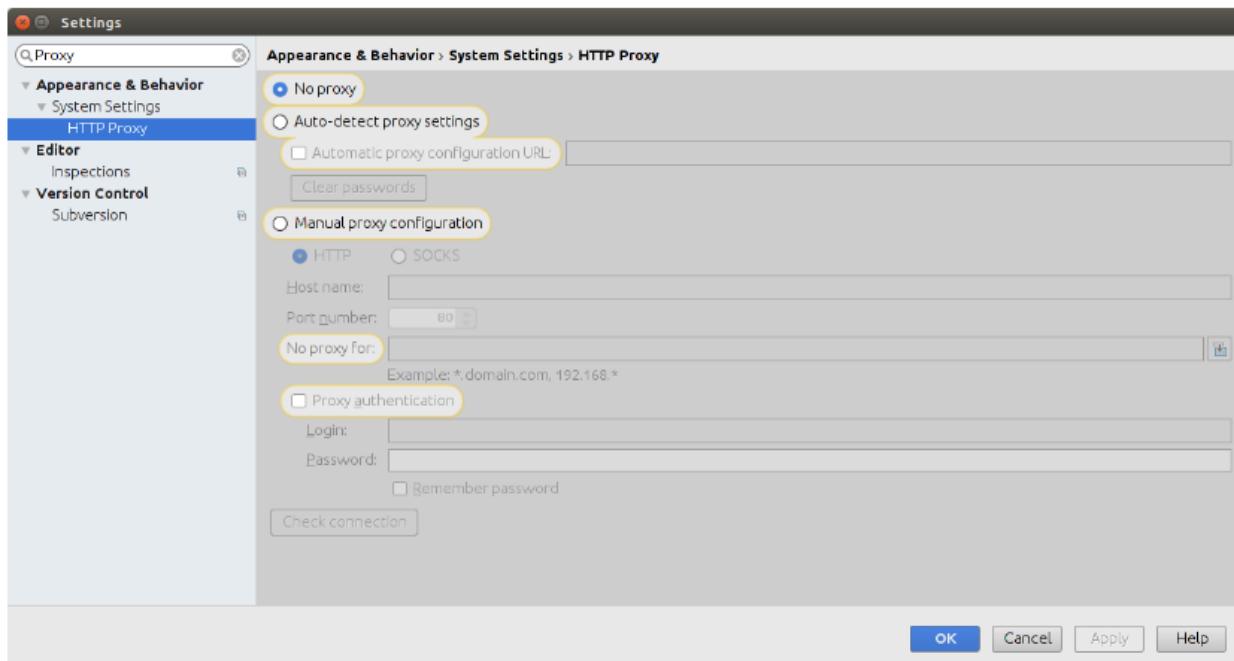
2. Dev channel :Dev build های build، نسخه های برتر و دستچین شده از build های قدیمی تر هستند که تست شده و مورد تایید می باشند. این نسخه هر دو هفته یا ماهی یکبار بروز آوری می شود.

- .3: نسخه هایی با سطح کیفیتی beta در این channel عرضه می شوند.
- .4: جدیدترین نسخه‌ی ثابت از Android Studio که تمامی خطاهای آن برطرف شده و آماده‌ی عرضه‌ی نهایی در اختیار توسعه دهنده و کاربران می باشد.



## 6-1-1- استفاده از proxy با Android Studio

می توانید Android و Gradle را با proxy را بکار ببرید. برای این منظور کافی است از مسیر... Android Studio > File > Settings... همچنین تنظیمات Gradle را برای استفاده از این proxy بروز رسانی می کند. از ویرایش 1.4 محیط کاری Android Studio این امکان بهینه شده است.



## 11-7-ابزار Android Device Monitor

جهت اجرای Android Device Monitor می توانید یکی از دو مسیر رو به رو را طی نمایید:  
Android Studio Tools > Android > Android Device Monitor  
بر روی Android Device Manager کلیک کنید.

در پی این کار، یک اپلیکیشن مبتنی بر Eclipse به نام Android Device Monitor برای تعامل با دستگاه مجازی اندروید و اپلیکیشن اندرویدی باز می شود.

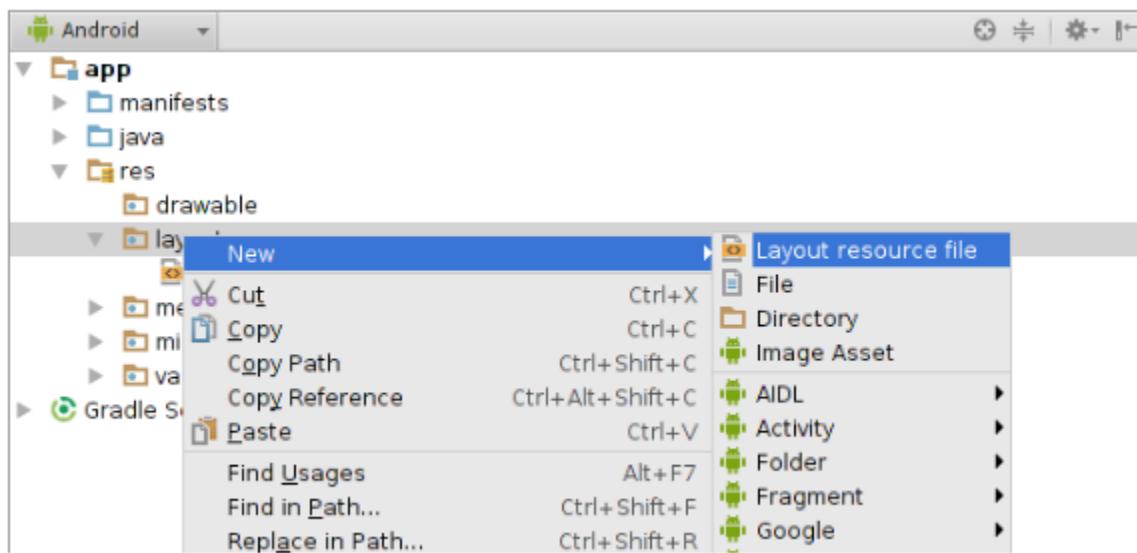
## 12-1-شروع به استفاده از Android Studio

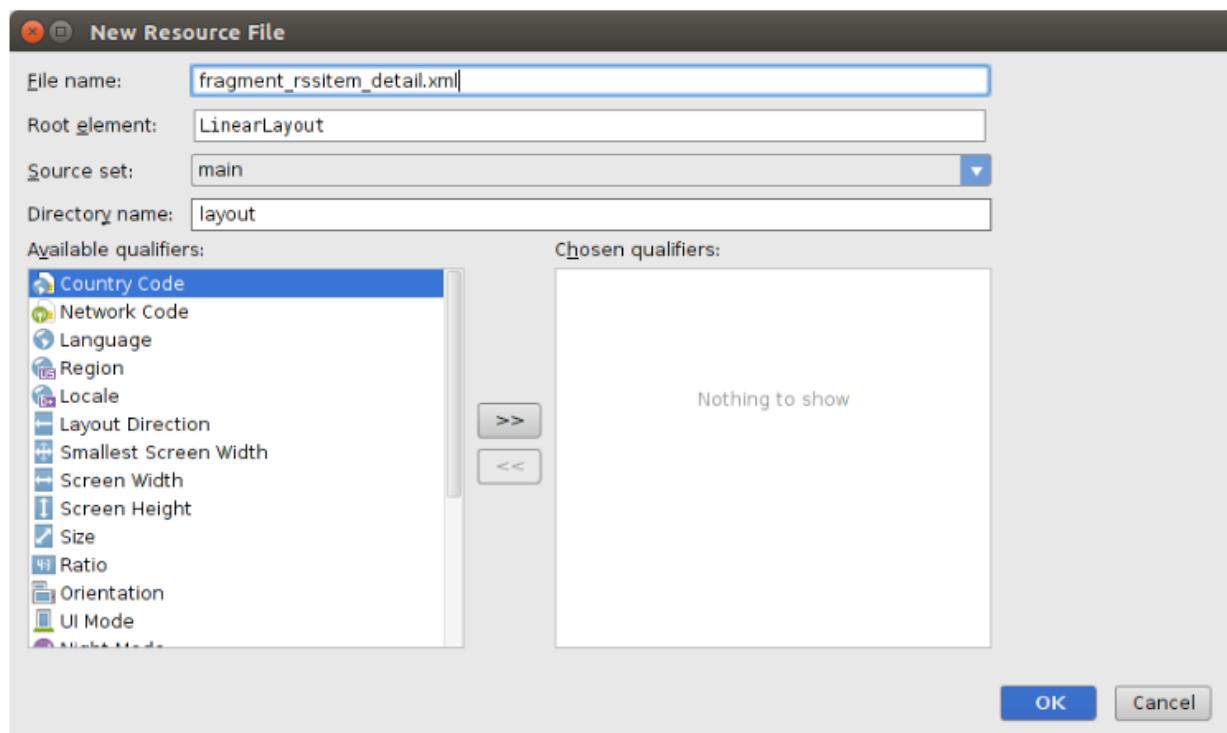
المان هایی که در منوهای Android Studio مشاهده می کنید پویا هستند و می توانند بر اساس انتخاب جاری کاربر تغییر کنند. اما متاسفانه این ویژگی در زمان حاضر هنوز کامل نیست و آن طور که باید عمل نمی کند. در صورتی که المان مورد نظر را مشاهده نکردید، سعی کنید دوباره بر روی المان app کلیک نمایید.

## 12-1- ایجاد یک فایل layout جدید

جهت ایجاد یک فایل layout جدید، پروژه را انتخاب نموده، بر روی آن راست کلیک نمایید و سپس این مسیر را طی کنید: File ▶ New ▶ Android resource file . با انتخاب گزینه ی Layout، یک فایل جدید XML برای تعریف المان های رابط کاربری خود (فایل Layout) ایجاد نمایید.

آسان ترین راه برای ایجاد یک فایل حاوی محتوا (resource file) layout همچون فایل layout که ظاهر برنامه را تعیین می کند، context menu است که با راست کلیک بر روی پوشه ی مربوطه به نمایش در می آید. نحوه ی ایجاد یک فایل layout جدید را در تصویر زیر مشاهده می کنید.

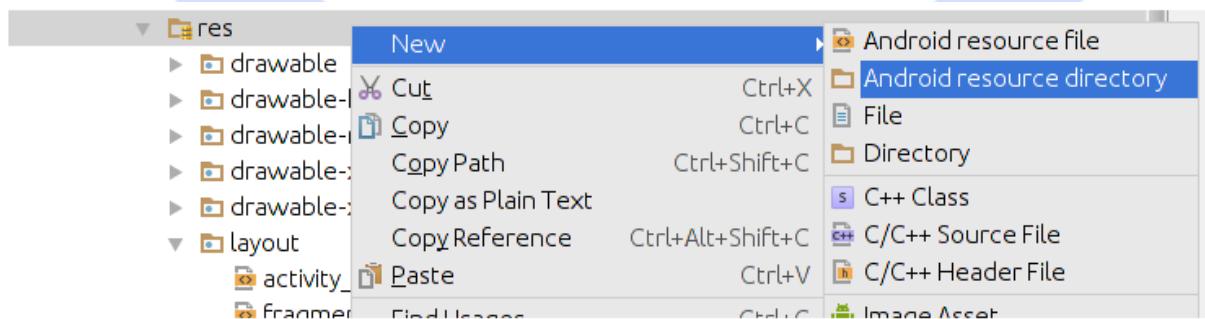


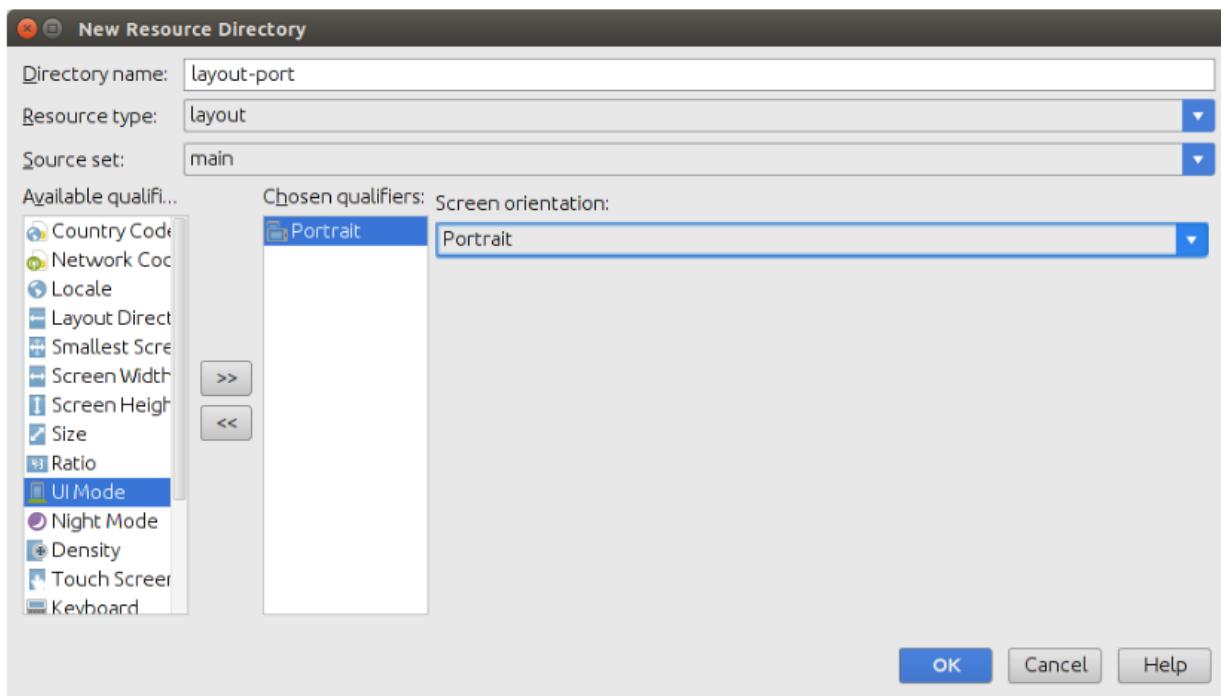


**نکته:** اگر می خواهید یک فایل جدید layout برای دستگاه با پیکربندی خاص طراحی کنید، در همین پنجره‌ی محاوره‌ی ای می توانید آن را از بخش Available Qualifier انتخاب نمایید.

## 2-12-1- ایجاد یک فایل محتوای جدید (resource file)

در صورتی که فایل محتوای مورد نظر موجود نبود، بایستی آن را ایجاد کنید. در تصویر زیر مشاهده می کنید که فایل محتوای layout-port\_folder ایجاد می شود.





### 12-1-3 کار با فایل های layout

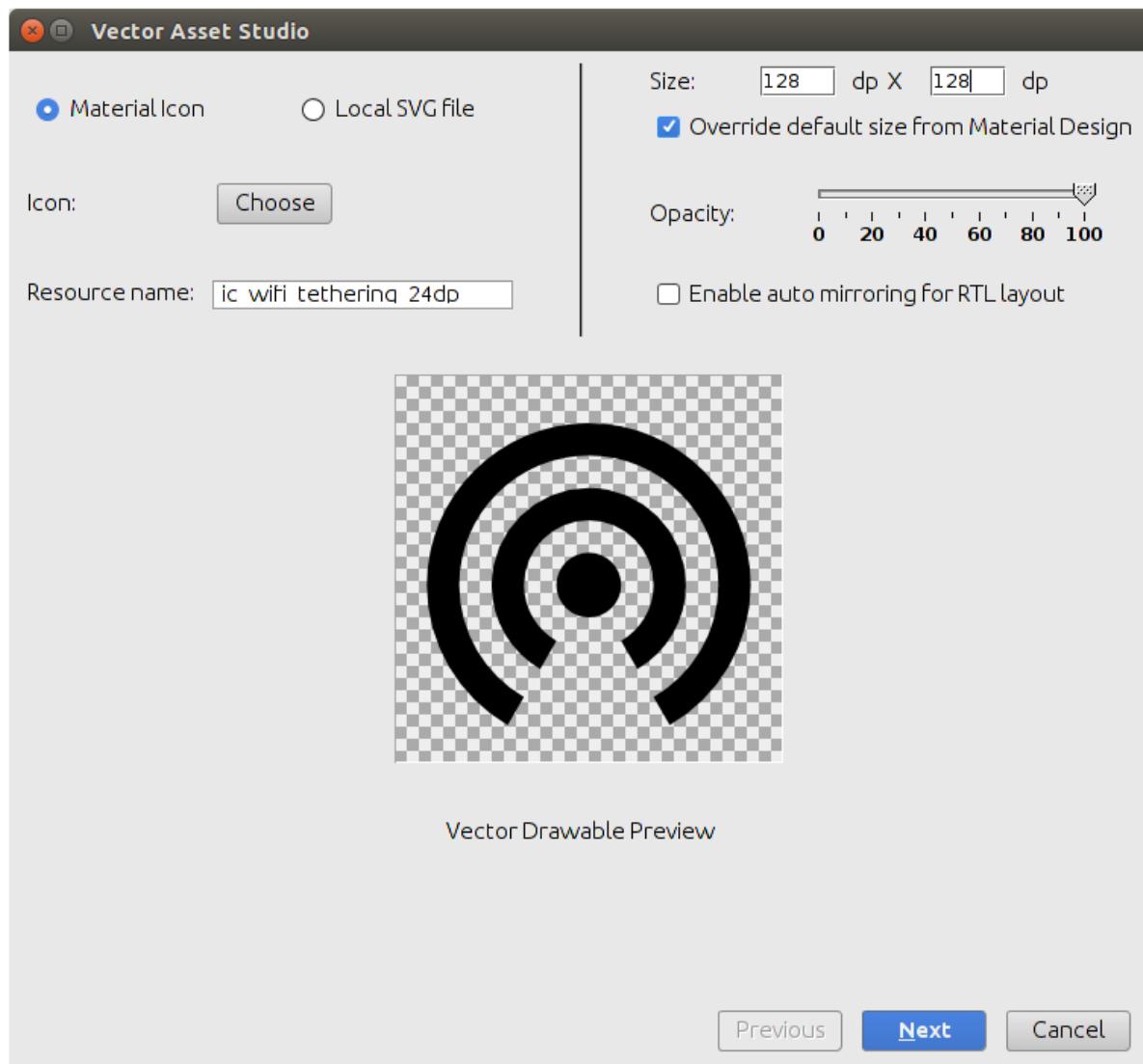
محیط کاری Android Studio یک ویرایشگر ویژوال و متنی برای اعمال تغییر در فایل های layout فراهم می آورد. آسان ترین روش برای افزودن ابزارک های رابط کاربری (UI Widget) به فایل layout، انتخاب آن ها از Palette است. می توانید المان های مورد نظر را با اشاره گر موس کشیده و به راحتی در فایل جایگذاری نمایید. در حالت طراحی ویژوال (visual design mode)، شما می توانید با راست کلیک بر روی view مورد نظر و انتخاب گزینه Context Delete از menu، آن view را حذف نمایید.

Android Studio مقادیر جای نگهدار و موقتی (placeholder) را با مقادیر حقیقی جایگزین می کند. این امر سبب می شود مقایسه های فایل ها کمی دشوار شود. در چنین موقعی برای پی بردن به مقادیر موقتی یا placeholder، کافی است بر روی مقادیر مورد نظر دوبار کلیک نمایید.

### 12-4-1 قرار دادن فایل تصویری (image) در پروژه

منوی محیط کاری Android Studio را باز نموده و سپس این مسیر را طی نمایید: File > New > Vector Asset. در پی این کار یک فایل drawable برداری یا vector ایجاد می

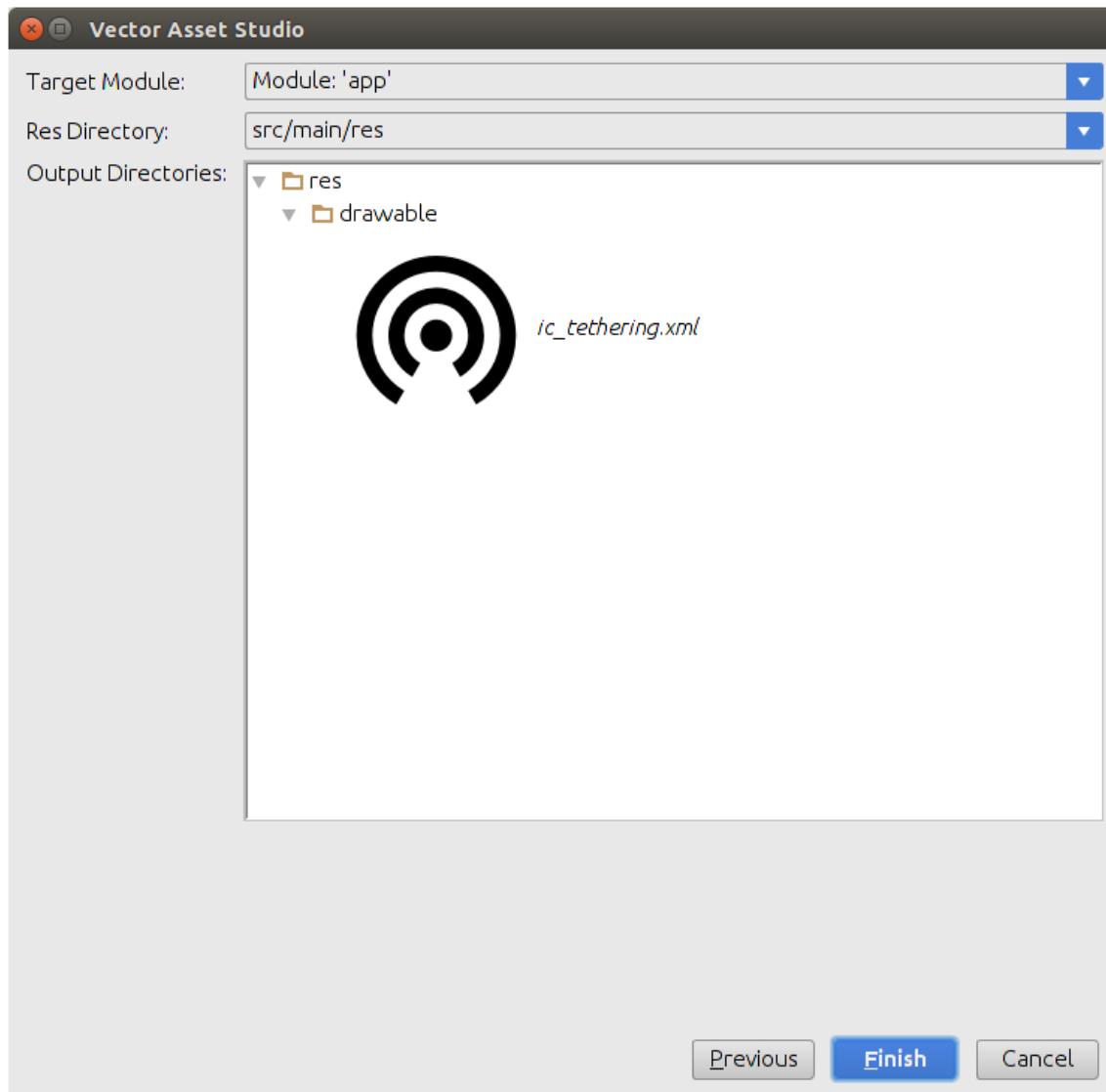
شود. در زیر پروسه‌ی ایجاد آیکون `ic_tethering` در محیط کاری Android Studio به نمایش گذاشته شده است:



122

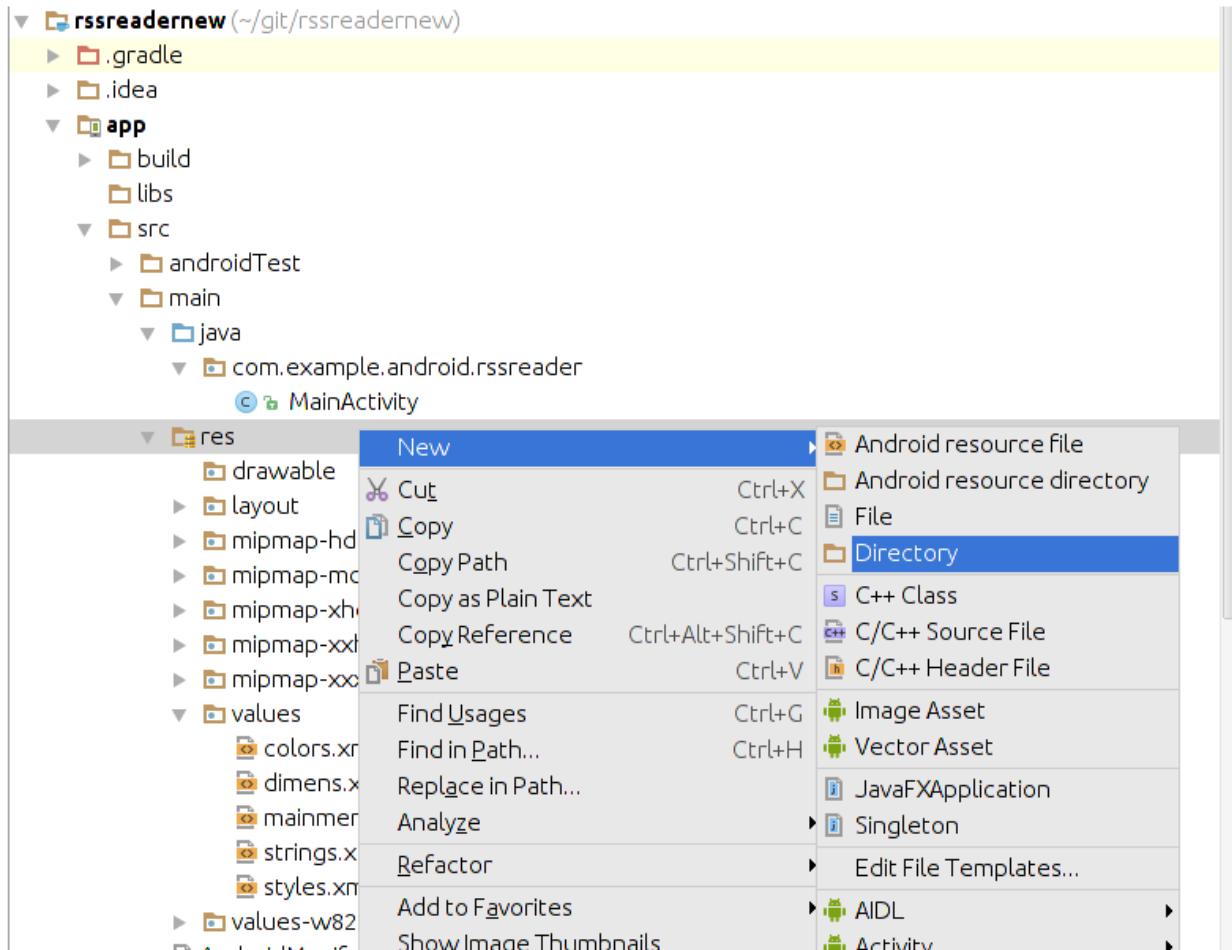
آدرس آموزشگاه: تهران - خیابان شریعتی - بالاتر از خیابان ملک - جنب بانک صادرات - پلاک 651 طبقه دوم - واحد 7

88146323 - 88446780 - 88146330

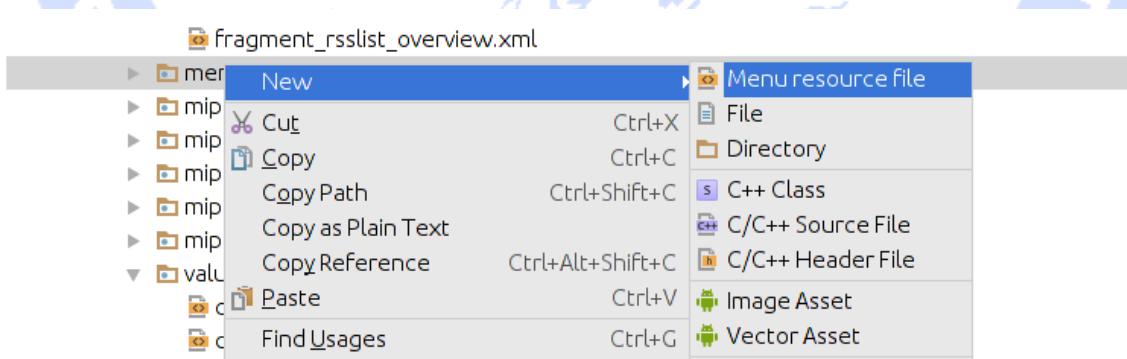


#### 1-12-4- افزودن یک منوی جدید (menu resource)

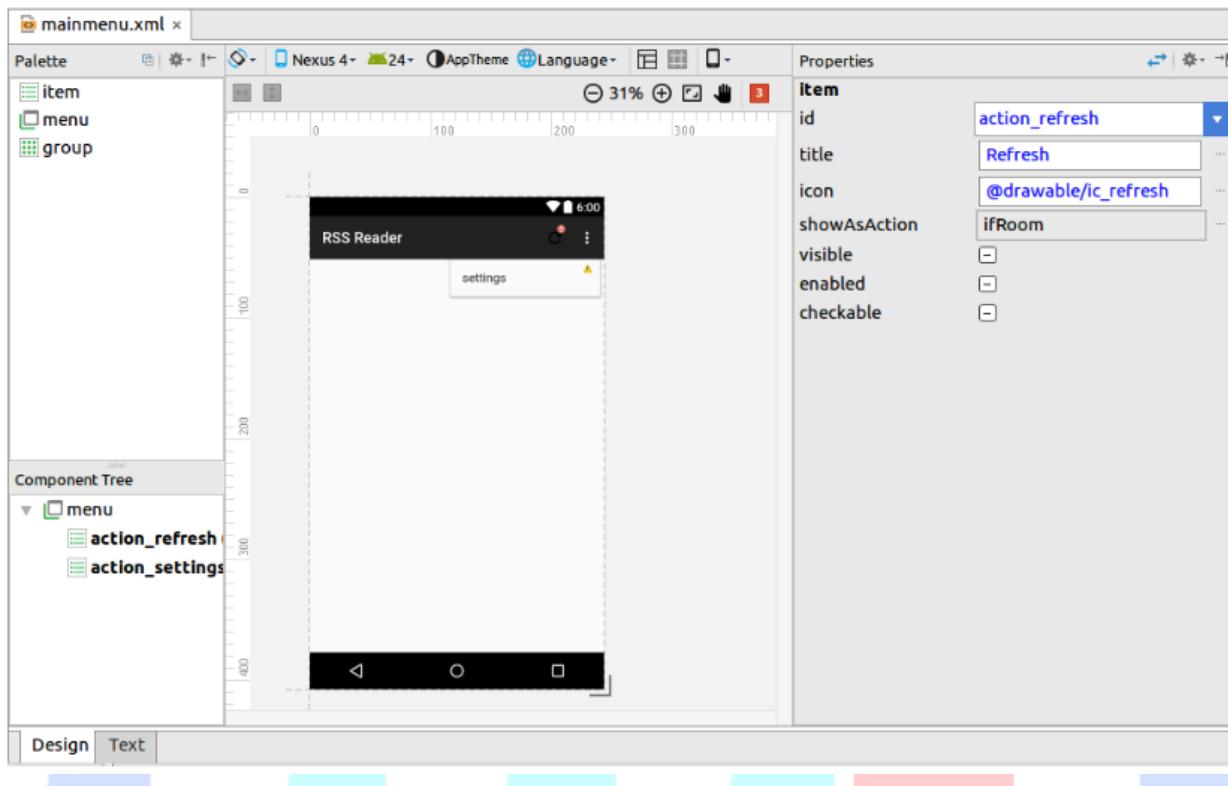
اگر پوشه ای به نام menu تعریف نشده، اکنون آن را ایجاد نمایید.



یک فایل XML جدید به نام mainmenu.xml برای این منو تعریف نمایید.

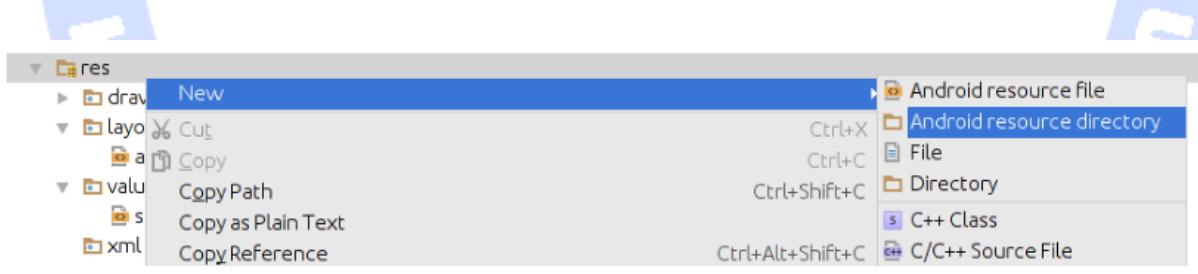


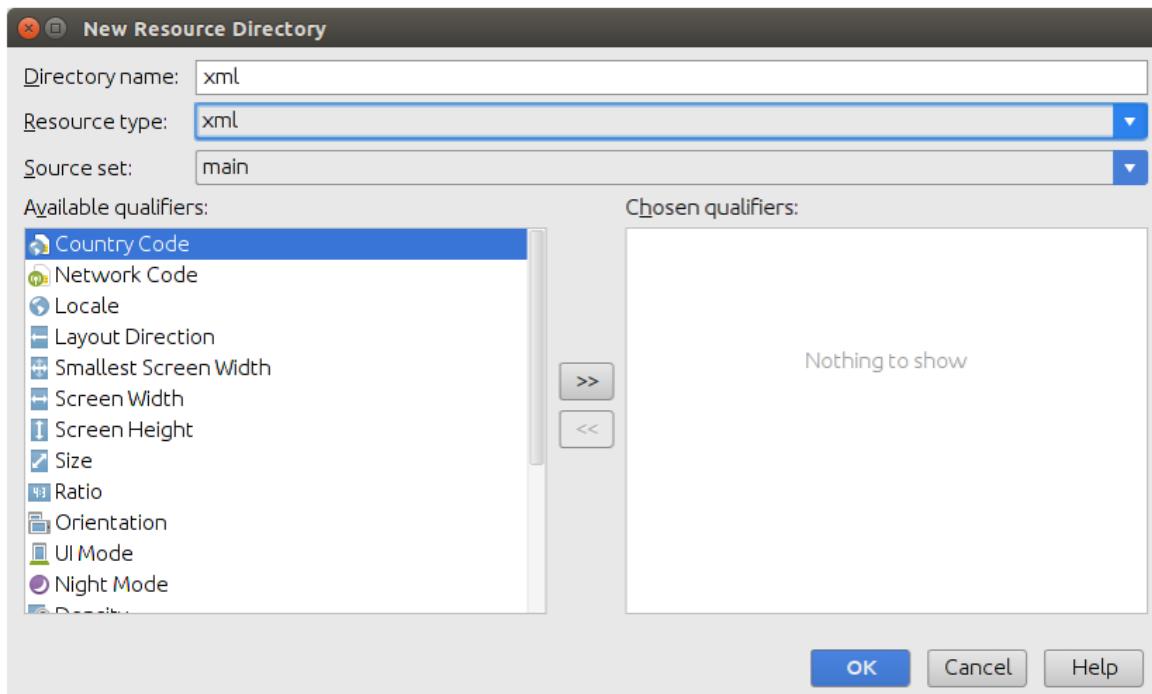
پس از آن می توانید فایل مزبور را از طریق تب Text به صورت دستی ویرایش کنید و یا تب Design را باز نموده و از طریق drag&drop محتوای فایل را ویرایش نمایید.



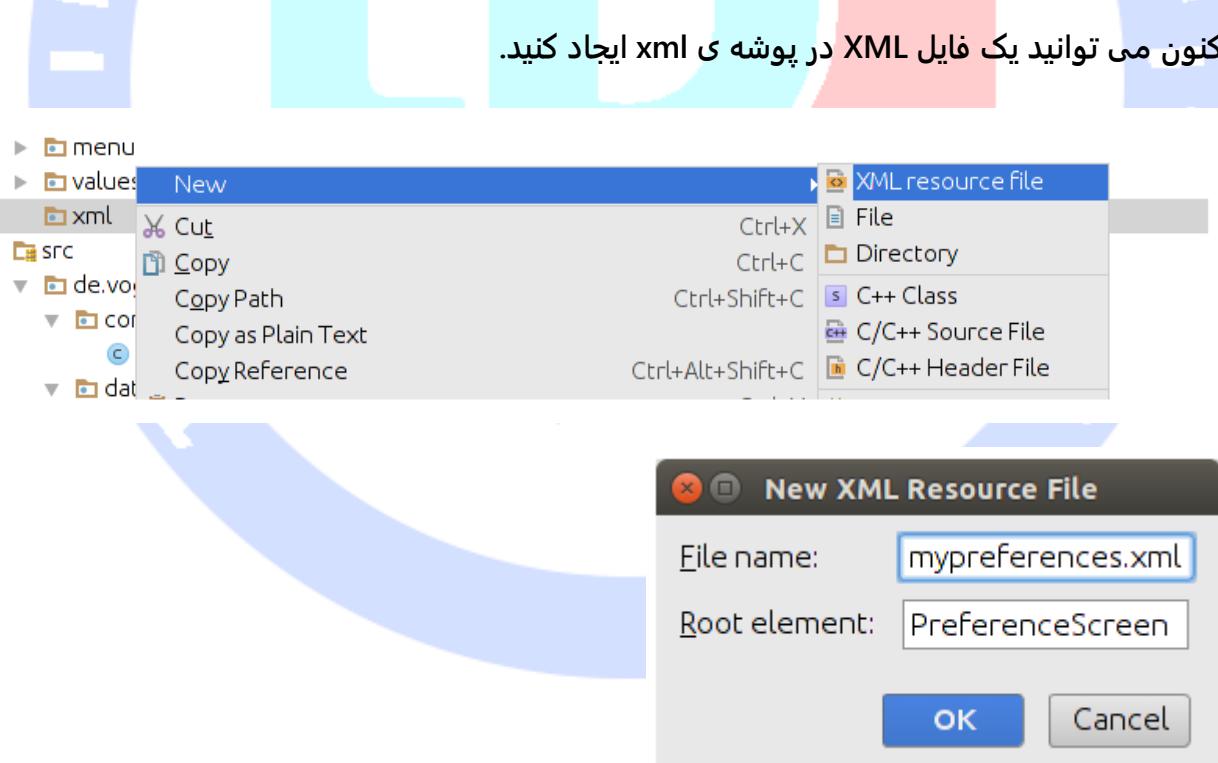
## 12-1- ایجاد فایل preference

اگر بروزه‌ی شما پوشه‌ی /app/res/xml/ ندارد، می‌بایست آن را ایجاد کنید.





اکنون می توانید یک فایل XML در پوشه‌ی xml ایجاد کنید.



## 12-7- مشاهده محتوای فایل های log اپلیکیشن اندرویدی

جهت مشاهده محتوای log و گزارشات ثبت شده از وقایع، کلیدهای Alt+6 را همزمان فشار دهید.

## 13-1- وظایف و task های مربوط به برنامه نویسی Java

### 13-1- ایجاد کد ... toString, Getter/Setter

جهت دریافت ساختمان یا اسکلت کد setter, getter و توابع سازنده (constructor) به صورت آماده از محیط، ابتدا منوی محیط کاری را باز نموده و سپس مسیر رو به رو را طی نمایید:

.Code ▶ Generate

### 13-2- استفاده از Java 8 در ساخت اپلیکیشن های اندرویدی

جدیدترین افزونه‌ی Gradle قادر به ترجمه و build کدهای Java 8 است و کاملاً با آن سازگار می‌باشد. از ویرایش Gingerbread (ورژن 9 کتابخانه‌های اندروید) به بعد می‌توانید از قابلیت‌ها و ابزار زیر استفاده نمایید:

1. عبارت‌های Lambda

2. Java.util.function

از ویرایش 7.0 اندروید به بعد می‌توانید از قابلیت‌ها زیر بهره بگیرید:

1. متدهای static و پیش فرض interface (متدهایی که توسط interface برای پیاده‌سازی در

کلاس در اختیار توسعه دهنده قرار می‌گیرد)

2. Repeatable annotation

3. Stream

4. Reflection API

جهت فعال سازی Java 8 در پروژه، تنظیمات زیر را بکار ببرید. بخش های مهم `jackOptions` و `compileOptions` هستند:

```
android {
    ...
    defaultConfig {
        ...
        jackOptions {
            enabled true
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
}
```

### 1-13-3- استفاده از live template و تکه کدهای آماده در کد برنامه‌ی خود

محیط کاری Android Studio برای task ها و وظایف معمول در java و android تکه کدهای آماده ارایه می دهد. لیست زیر پرکاربردترین آن ها را با ذکر کاربرد هر یک، عنوان کرده است:

.1 - یک Toast جهت نمایش پیغامی مختصر به صورت زودگذر برای کاربر، تعریف می کند.

.2 - متدها `findViewById` را همراه با تبدیل در اختیار شما قرار می دهد.

.3 - یک ثابت یا constant تعریف می کند.

به منظور مشاهده ای لیست کاملی از template های یاد شده، این مسیر را طی نمایید:  
کارکردن با live template  
.File ▶ Settings ▶ Editor ▶ Live Templates

### 1-14-1- کار با نرم افزار کنترل نسخه‌ی Git

#### 1-14-1- Git repository یا کپی کردن یک Clone

برای ایجاد یک کپی محلی از repository مورد نظر، این مسیر را طی نمایید:

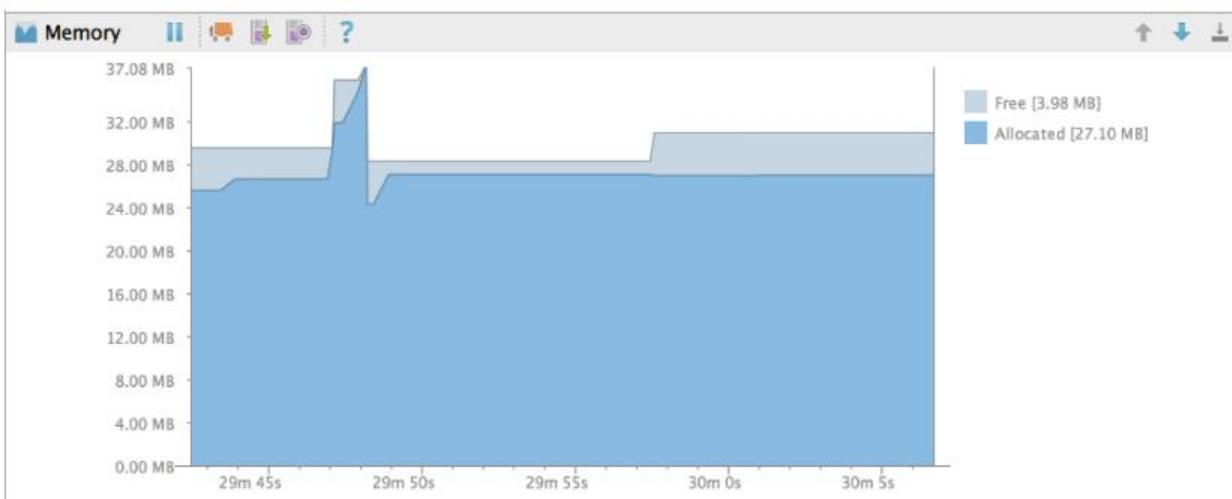
Version Control ▶ Git

## 14-1-کپی کردن یک repository از Github

Version Control ▶ Github ▶ Checkout

## 15-1-ناظارت بر مصرف حافظه (Memory Monitor)

به منظور مشاهده ی میزان حافظه ای که برنامه مصرف می کند، ابتدا باید اپلیکیشن را اجرا نموده و حین اجرای آن، Android Monitor را باز کنید. پس از کلیک بر روی تب `Monitors`، `Memory` را باز نمایید.



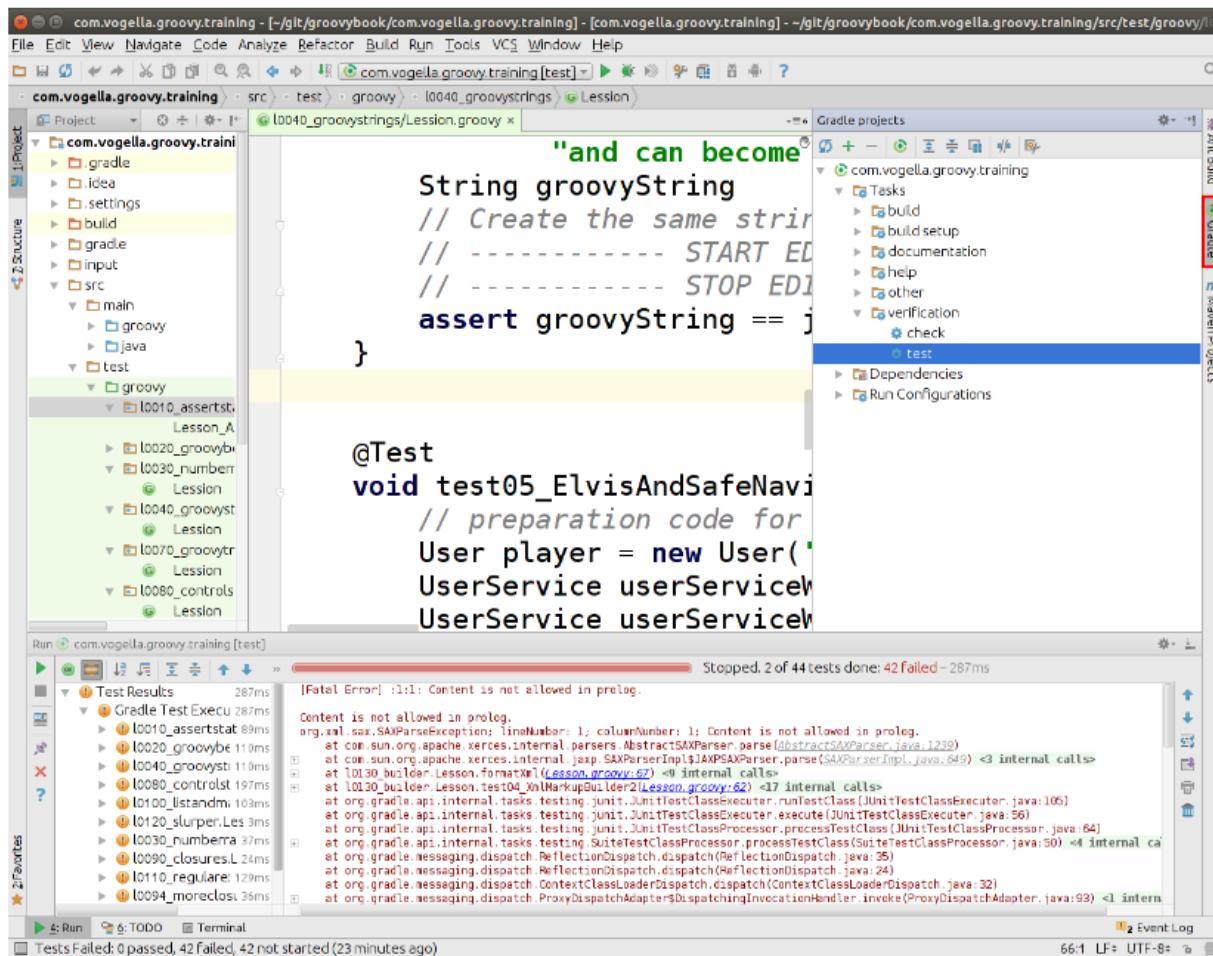
## 16-1-استفاده از Gradle در محیط برنامه نویسی IntelliJ

16-1-محیط توسعه‌ی IntelliJ و قابلیت پشتیبانی از Gradle  
محیط IntelliJ با تنظیمات پیش فرض خود به راحتی از Gradle پشتیبانی می کند.

### Gradle project view-1-16-2

برای نمایش .View ▶ Tool Buttons، این مسیر را طی نمایید:

با این کار یک نوار ابزار عمودی در سمت راست محیط به نمایش در می آید. با کلیک بر روی دکمه `Gradle Projects`، `Gradle Projects` را باز نمایید.



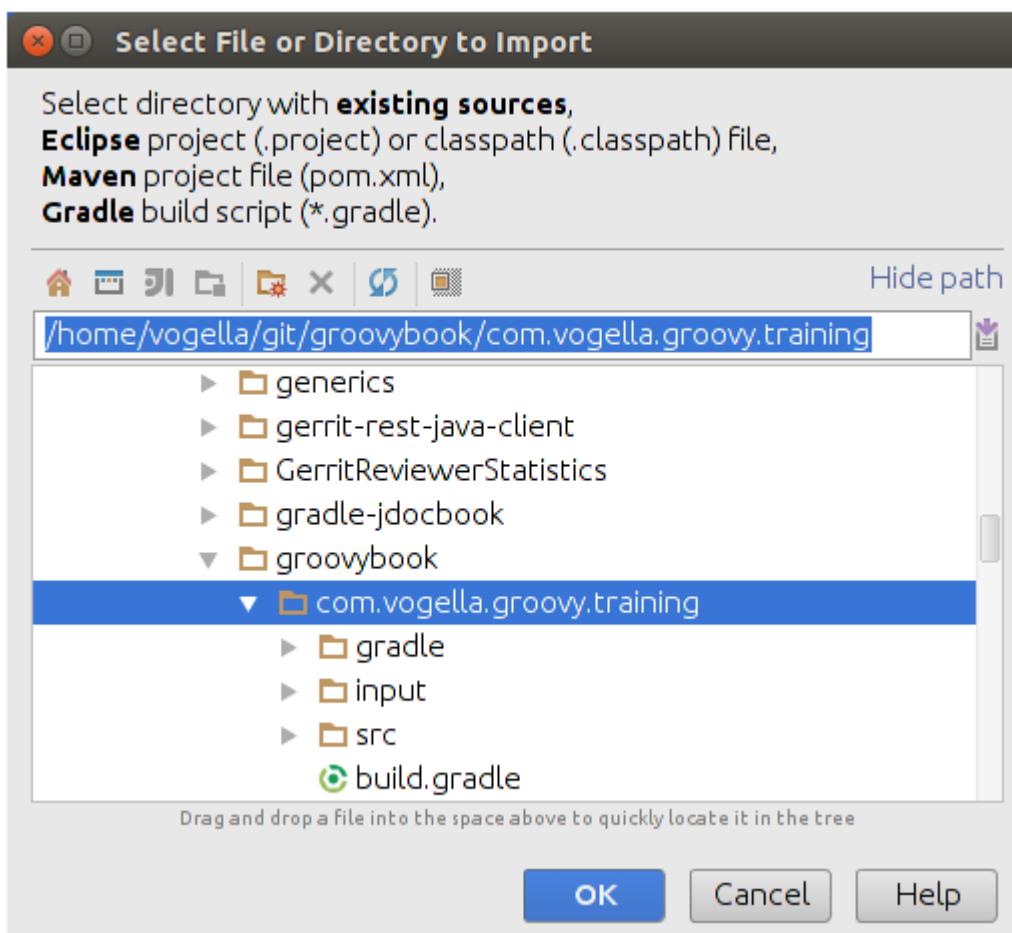
### 3-1-16-3- راه اندازی task های Gradle

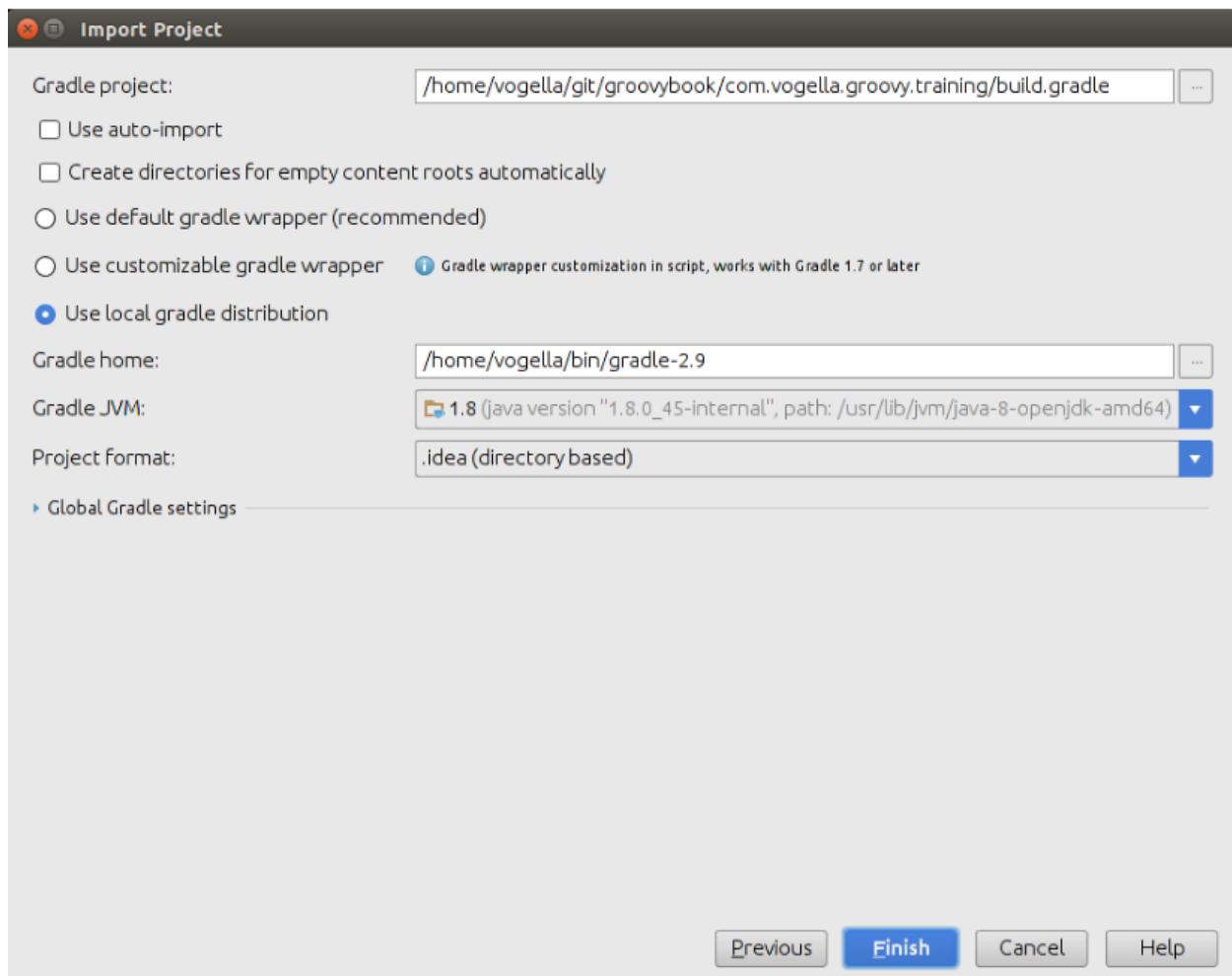
بر روی یکی از task ها در Gradle Projects کلیک نمایید تا اجرا شود.

### 3-1-16-4- وارد کردن آماده‌ی (import) یک پروژه‌ی آماده‌ی (import) در محیط برنامه نویسی IntelliJ

پروژه‌ی جاری را بسته (File ▶ Close Project) و سپس بر روی لینک Import Project کلیک نمایید.  
پس از آن مراحلی که در تصاویر زیر نمایش داده شده را به ترتیب دنبال کنید.

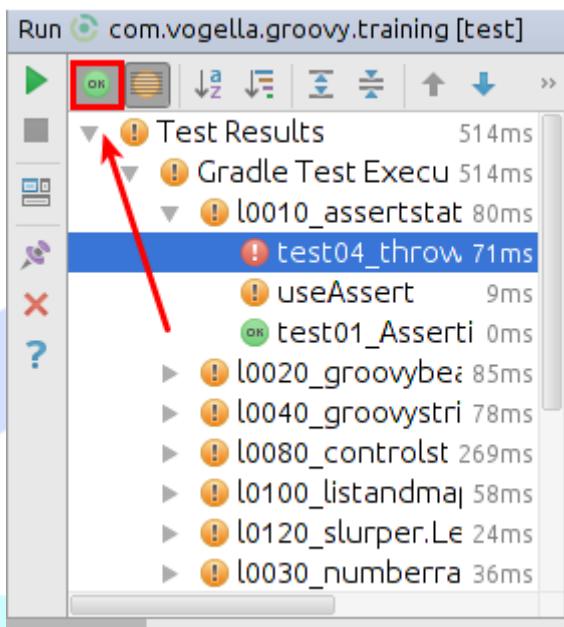






### ۱-۱۶-۵ مشاهده ی نتیجه ی تمامی تست های اجرا شده

بر روی دکمه ی اشاره شده در تصویر زیر کلیک کرده تا نتیجه ی تمامی تست ها (علاوه بر تست های ناموفق) به نمایش درآید.



## 17-1-انتقال پروژه از محیط Eclipse به Android Studio

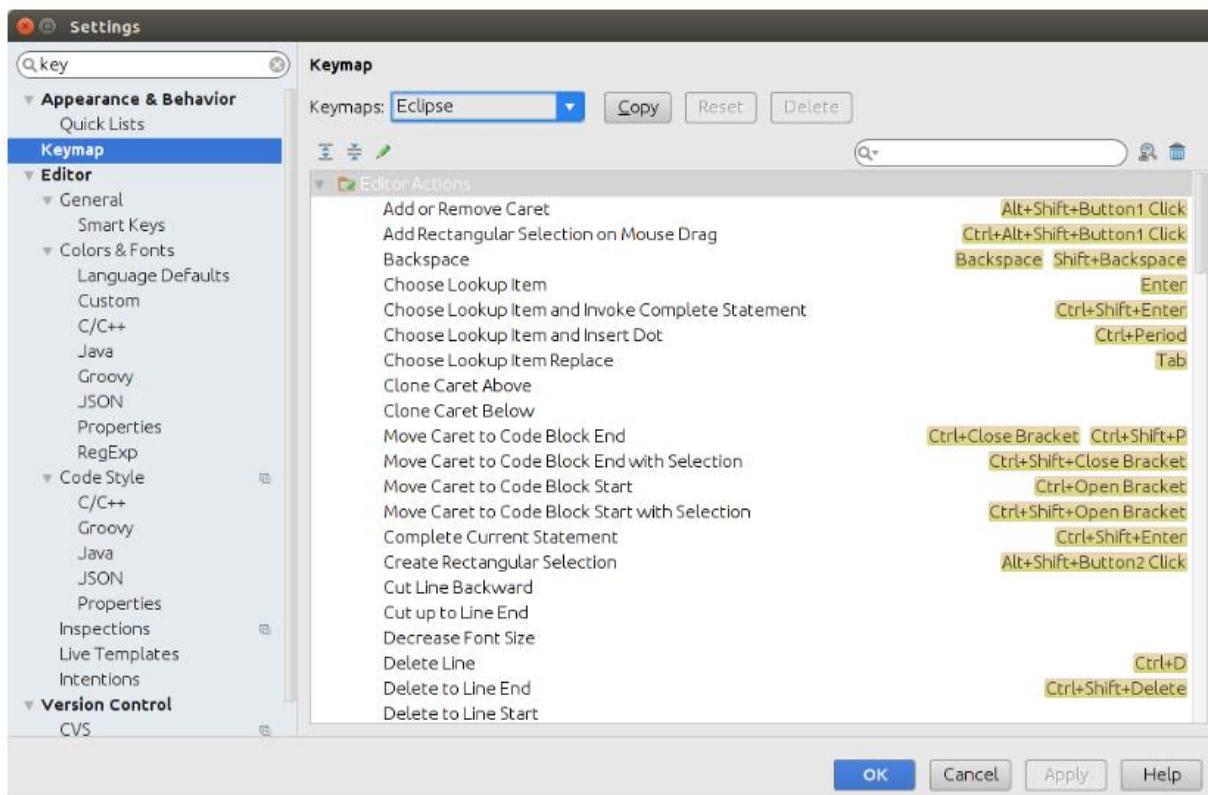
به منظور انتقال پروژه از محیط کاری Eclipse به Android Studio این مراحل را به ترتیب دنبال نمایید:

1. داخل محیط Android Studio، تمامی پروژه های باز را ببندید. به دنبال این کار صفحه **Import Non-Android Welcome** به نمایش در می آید. در این صفحه بر روی لینک **Studio project** کلیک نمایید.
2. به پوشه **کلید** رفته و دکمه **OK** را کلیک نمایید.

## 17-1-ویرایش keybinding های binding

### (تغییر تنظیمات صفحه کلید)

برای اینکه سرعت کار شما در محیط جدید افزایش یابد، ابتدا keybinding اندروید استودیو را به محیط کاری Eclipse تغییر دهید.





## 1-2-آموزش Intent در Android

این مبحث به آموزش نحوه‌ی استفاده از Intent جهت برقراری ارتباط بین کامپوننت‌ها و اجزا نرم افزاری مختلف اندروید می‌پردازد.

**1-2-1-اجرا و راه اندازی دیگر کامپوننت‌ها و اجزا نرم افزاری اندروید به وسیله‌ی intent‌ها (تبادل اطلاعات بین اجزا و کامپوننت‌های مختلف به وسیله‌ی آبجکت‌های intent)**

Intent‌ها پیغام‌هایی هستند که به واسطه‌ی آن‌ها از سیستم تقاضای انجام کار یا عملیات خاصی را می‌کنند. در واقع با استفاده از آبجکت intent می‌توانید بین کامپوننت‌های مختلف یک اپلیکیشن و حتی کامپوننت‌های نرم افزاری دیگر اپلیکیشن‌ها تعامل برقرار نمایید و اطلاعات رد و بدل کنید. برای مثال، یک Activity می‌تواند دیگر را جهت عکس گرفتن اجرا کند.

به وسیله‌ی intent‌ها می‌توانید service‌ها و Activity‌ها را اجرا کنید یا مقادیری را از یک activity به activity دیگری ارسال نمایید.

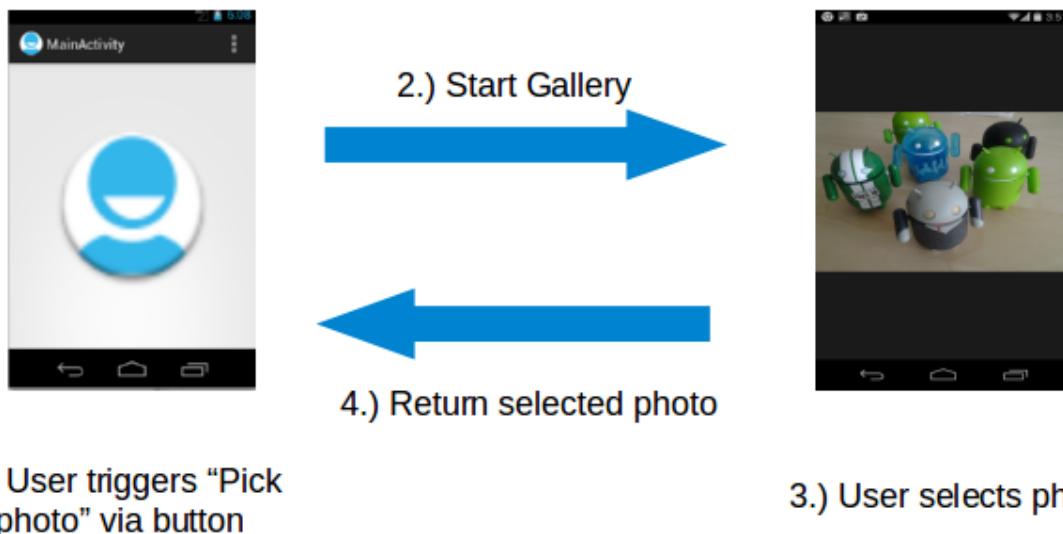
Intent‌ها آبجکت یا نمونه‌هایی از جنس کلاس android.content.Intent هستند. کد شما می‌تواند آبجکت‌های intent را به سیستم اندروید ارسال کرده و صریحاً اعلام کند کدام کامپوننت‌ها هدف و مد نظر شما هستند. به طور مثال، می‌توانید یک activity را به کمک متود startActivity() راه اندازی کنید (تعریف کنید که این intent باید فقط به قصد اجرای activity بکار گرفته شود).

داده‌های مورد نظر را می‌توانید برای ارسال به activity دیگر داخل Bundle قرار دهید. داده‌هایی که داخل این آبجکت قرار دادید توسط کامپوننت مد نظر دریافت و مورد استفاده قرار می‌گیرد.

در سیستم اندروید استفاده‌ی مجدد از دیگر کامپوننت‌های اپلیکیشن تحت عنوان task شناخته می‌شود. یک اپلیکیشن می‌تواند برای انجام عملیات خاصی به دیگر کامپوننت‌های اندروید دسترسی داشته باشد. به عنوان مثال می‌توانید از یک کامپوننت اپلیکیشن خود، کامپوننت دیگری در سیستم عامل اندروید را فعال سازی نمایید که فایل‌های تصویری را مدیریت می‌کند، با اینکه

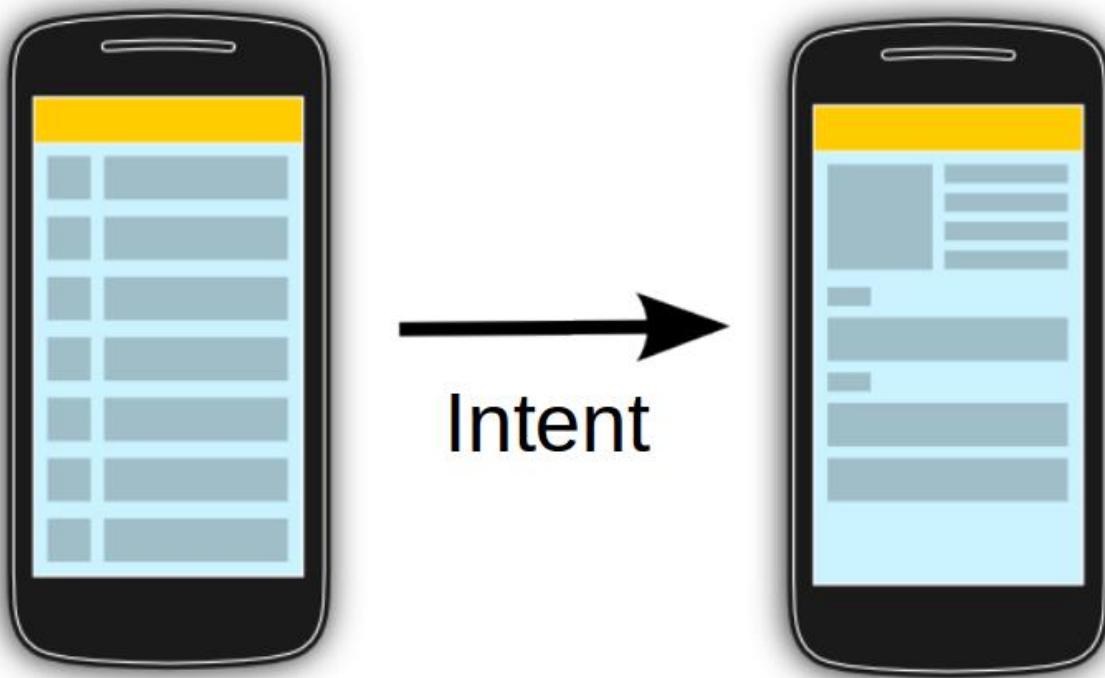
این کامپوننت به اپلیکیشن جاری تعلق ندارد. سپس در این کامپوننت عکس دلخواه را انتخاب کرده و به اپلیکیشن خود برای استفاده از عکس انتخابی باز گردید.

این این جریان تعامل و تبادل اطلاعات در زیر به تصویر کشیده شده است.



## ۱-۲-۲- راه اندازی Activity ها به وسیله آبجکت intent

به منظور راه اندازی یک activity، می توانید متده استartActivity(intent) را بکار ببرید. این متده در سطح آبجکت Context تعریف شده است. کلاس Activity از این کلاس پدر ارث بری داشته و متعاقباً متده ذکر شده را نیز از این کلاس به ارث می برد و آن را در بدنه ای خود فراخوانی می کند.



کد زیر نحوه‌ی راه اندازی یک activity از طریق آبجکت intent را به نمایش می‌گذارد:

```
# Start the activity connect to the
# specified class
Intent i = new Intent(this, ActivityTwo.class);
startActivity(i);
```

Activity هایی که توسط activity های دیگر راه اندازی می‌شوند در اصطلاح sub-activity خوانده می‌شوند.

### ۱-۲-۲- راه اندازی service ها

گفتنی است که توسط intent ها می‌توان service ها را نیز راه اندازی نمود. کافی است متده فراخوانی startService(Intent) را فراخوانی نمایید.

## ۱-۲-۴-۱-۲-ارسال intent های صريح/explicit و ضمني/implicit

در اندرويد دو نوع intent وجود دارد. intent صريح و intent ضمني.

Intent صريح يا explicit برای صدا زدن و به اجرا در آوردن service یا activity مشخص مورد استفاده قرار می گيرد. در اين سناريyo کاربر نمي تواند تصميم بگيرد رخداد مورد نظر توسيط کدام کامپوننت انجام شود، برای مثال شرایطي را در نظر بگيريد که برنامه ی کاربردي شما دو activity دارد و شما می خواهيد از طريق activity اول به activity دوم راه پيدا کنيد. در اين حالت لازم است از intent صريح استفاده نمایيد. به عبارت ديگر به منظور برقراری ارتباط بين بخش هاي داخلی يك نرم افزار از intent صريح استفاده می شود. Intent های ضمني هدف و کامپوننت مد نظر خود را با نام مشخص نمی کنند و معمولا برای راه اندازی کامپوننت ديگر نرم افزار ها مورد استفاده قرار می گيرند. به بيان ديگر زمانی که می خواهيد سیستم اندرويد عملیاتی را انجام دهد و برای شما چندان اهمیتی ندارد این کار توسيط کدام activity یا service انجام می شود، می بايست از intent ضمني استفاده نمایيد. در اين سناريyo سیستم اندرويد تمامی کامپوننت هایی که قابلیت انجام درخواست را دارند، فهرست می کند و کاربر می تواند تصميم بگيرد درخواست توسيط کدام کامپوننت اجرا شود.

کد زير نحوه ی ايجاد يك intent صريح و ارسال آن به سیستم Android جهت راه اندازی activity دیگر را نمایش می دهد.

```
Intent i = new Intent(this, ActivityTwo.class);
i.putExtra("Value1", "This value one for ActivityTwo ");
i.putExtra("Value2", "This value two ActivityTwo");
```

همان طور که در بالا ذکر شد، هنگامی که intent درخواست انجام عملیات خاصی را از سیستم اندرويد می کند، اندرويد تمامی نرم افزارهایی که برای اجرای این درخواست و عملیات ثبت شده اند را شناسایی کرده و برای کاربر فهرست می کند. این کار توسيط intent filter صورت می پذيرد. برای مثال، کد زير از اندرويد درخواست مشاهده ی يك صفحه ی وب را می کند. تمامی مرورگرهای وب که برای اين منظور ثبت شده اند، توسيط اندرويد شناسایی شده و برای کاربر جهت انتخاب، نمایش داده می شود. اين کار توسيط intent filter امكان پذير می باشد.

کد زیر به سیستم اندروید اعلام می کند که یک صفحه ی وب را باز کند. تمامی مرورگر های نصب شده بر روی سیستم که برای این intent ثبت شده اند، لیست می شوند و کاربر یکی از آن ها را برای مشاهده ی صفحه انتخاب می کند.

```
Intent i = new Intent(Intent.ACTION_VIEW, Uri.parse("http://www.vogella.com"));
startActivity(i);
```

## 2-2-انتقال و تبادل اطلاعات بین activity ها و service ها

### 2-2-ارسال داده به کامپونت مقصد

در زمان تعریف یک intent باشتنی اطلاعات مرتبط با آن intent را نیز مشخص نمایید. در حقیقت یک intent می تواند دربردارنده ی یک سری اطلاعات پایه ای پیرامون خود باشد همچون اینکه چه عملیاتی باید انجام شود و چه نوع و غیره ... برای هر نوع درخواست نیز ممکن است اطلاعات بیشتری نیاز باشد. یک intent می تواند همچنین داده های اضافی در نمونه ای از کلاس (آبجکت) Bundle داشته باشد که به راحتی از طریق متدهای getExtras() قابل بازیابی است.

شما می توانید داده های اضافی را مستقیماً از طریق نسخه های overload شده می تند از آبجکت های Intent، (نمونه ای از کلاس) Bundle قرار دهید. داده های اضافی در قالب جفت های کلید/مقدار هستند. کلید همیشه از نوع String است و مقادیر آن را می توانید از نوع داده ای اولیه int، float، String، Serializable و Parcelable یا آبجکت های از نوع Bundle یا آبجکت های آبجکت (آبجکت های آبجکت) تنظیم نمایید.

کامپونت دریافت کننده می تواند به این اطلاعات از طریق توابع `getAction()` و `getIntent()` آبجکت دسترسی داشته باشد. خود آبجکت Intent را نیز می توان از طریق متدهای `getIntent()` بازیابی نمود.

کامپوننتی که دریافت کننده ی این intent می تواند با فراخوانی `getIntent().getExtras()` به داده های اضافی دسترسی داشته باشد. این عملیات در تکه کد زیر به نمایش گذاشته شده است.

```

Bundle extras = getIntent().getExtras();
if (extras == null) {
    return;
}
// get data via the key
String value1 = extras.getString(Intent.EXTRA_TEXT);
if (value1 != null) {
    // do something with the data
}

```

## نمونه: استفاده از intent جهت به اشتراک گذاری اطلاعات

اغلب اپلیکیشن های اندروید این امکان را به شما می دهد تا داده هایی را با کاربران که عضو اپلیکیشن های Facebook, G+, Twitter و Gmail هستند به اشتراک بگذارید. در واقع می توانید داده های مورد نظر را از طریق آبجکت Intent به یکی از این کامپوننت ها ارسال کنید. تکه کد زیر روش استفاده از intent جهت به اشتراک گذاری اطلاعات را نمایش می دهد:

```

// this runs, for example, after a button click
Intent intent = new Intent(Intent.ACTION_SEND);
intent.setType("text/plain");
intent.putExtra(android.content.Intent.EXTRA_TEXT, "News for you!");
startActivity(intent);

```

## 2-2-2- بازیابی اطلاعات از یک subactivity

یک activity را می توانید با اعمال دکمه ی بازگشت در نمایشگر گوشی خود پایان دهید. با فشردن این دکمه، متدهای finish() فراخوانی می شود. چنانچه activity مورد نظر با صدا خوردن متدهای startActivity(Intent) اجرا شده باشد، در آن صورت فراخواننده انتظار بازگشت هیچ نتیجه یا بازخوردی از activity مزبور را ندارد.

اگر activity را با فراخوانی متدهای startActivityForResult() راه اندازی کرده باشید، در آن صورت طبق انتظار، نتیجه ای را بر می گرداند. با پایان یافتن subactivity، متدهای onActivityResult() در subactivity فرخوانده می شود و شما می توانید عملیاتی را با توجه به نتیجه بازگشتی انجام دهید.

در فراخوانی متدهای `startActivityForResult()` شما می‌توانید کدی (resultcode) جهت شناسایی راه اندازی شده تعریف نمایید. این کد به شما بازگردانده می‌شود. activity فراخوانده شده نیز می‌تواند کدی داشته باشد که فراخواننده به کمک آن می‌تواند تشخیص دهد آیا activity لغو شده یا خیر.



با استفاده از `finish()` یک intent جدید ایجاد کرده و داده ها را در آن می‌ریزد. سپس به واسطه ای فراخوانی متدهای `setResult()` نتیجه را تنظیم می‌کند.

نمونه کد زیر نشان می دهد چگونه با اجرای متده استartActivityForResult() و ارسال کلاس ایجاد شده از intent به عنوان آرگومان به این متده، یک intent را فعال و به اصطلاح trigger نمایید.

```
public void onClick(View view) {
    Intent i = new Intent(this, ActivityTwo.class);
    i.putExtra("Value1", "This value one for ActivityTwo ");
    i.putExtra("Value2", "This value two ActivityTwo");
    // set the request code to any code you like,
    // you can identify the callback via this code
    startActivityForResult(i, REQUEST_CODE);
}
```

زمانی که متده startActivityForResult() را صدا می زنید، راه اندازی شده sub-activity را از طریق آبجکت intent به فراخواننده خوانده می شود.

هنگامی که subactivity بسته می شود، به دنبالش داده ها را از طریق آبجکت intent به فراخواننده خود ارسال می کند. این عملیات داخل بدنه `i().finish()` پیاده سازی می شود.

```
@Override
public void finish() {
    // Prepare data intent
    Intent data = new Intent();
    data.putExtra("returnKey1", "Swinging on a star.");
    data.putExtra("returnKey2", "You could be better than you are.");
    // Activity finished ok, return the data
    setResult(RESULT_OK, data);
    super.finish();
}
```

با بسته شدن یا به پایان رسیدن subactivity، متده onActivityResult() در سطح کلاس activity فراخواننده صدا خورده می شود.

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (resultCode == RESULT_OK && requestCode == REQUEST_CODE) {
        if (data.getStringExtra("returnKey1")) {
            Toast.makeText(this, data.getStringExtra("returnKey1"),
                Toast.LENGTH_SHORT).show();
        }
    }
}
```

## 3- ثبت و معرفی intent filter برای intent filter

### 2-3-1 Intent filter چیست؟

Intent ها رخداد یک اتفاق را به سیستم اندروید اعلان می کنند و در پی آن تمامی Activity و Service هایی که برای آن اتفاق در سیستم ثبت شده اند، صدا خورده و اجرا می شوند. ها معمولاً عملیاتی که باید اجرا شوند را شرح داده و اطلاعاتی را نیز درباره ای عملیاتی که باید اجرا شود ارائه می دهد. برای مثال، اپلیکیشن شما می تواند جهت دسترسی به آدرس URL خاص، با استفاده از آبجکت intent یک کامپوننت مرورگر را راه اندازی کند. در تکه کد زیر این عملیات به نمایش گذاشته شده است.

```
String url = "http://www.vogella.com";
Intent i = new Intent(Intent.ACTION_VIEW);
i.setData(Uri.parse(url));
startActivity(i);
```

اینجا یک سوال مطرح می شود: چگونه سیستم اندروید می تواند کامپوننت هایی را که قادر به اجرای درخواست intent خاص و واکنش نشان دادن به آن هستند را شناسایی کند؟

در سیستم اندروید، کامپوننت ها می توانند به وسیله ای intent filter خود را برای عملیات یا داده ای خاص ثبت کنند و در واقع اعلان نمایند که قابلیت انجام درخواست و عملیات intent مورد نظر را دارند. به بیانی دیگر intent filter با تعریف قابلیت های یک کامپوننت، نوع intent ای که یک broadcast receiver یا service، activity قادر به پاسخ گویی آن است را مشخص می کند.

کامپوننت های اندروید intent filter را یا به صورت static و ثابت در فایل AndroidManifest.xml تعریف می کند و یا در خصوص broadcast receiver به صورت dynamic و از طریق کدنویسی. تعریف intent filter از طریق action، category و data می شود (در فایل تنظیمات به وسیله ای `<intent-filter>` تگ می توانید فهرستی از action، category، data هایی را که با هر کدام از کامپوننت های broadcast receiver یا service، activity مرتبط هستند را معرفی کنید). هر یکی از intent filter می تواند meta data می تواند data می داشته باشد.

زمانی که یک intent به سیستم اندروید ارسال می شود، محیط (platform) یا بستر اجرای android دریافت کننده ی آن را شناسایی می کند. این کار را با استفاده از داده های موجود در آبجکت intent انجام می دهد. در صورتی که چندین کامپوننت برای یک intent filter ثبت نام کرده و گوش بدھند، در آن صورت سیستم اندروید لیستی از کامپوننت هایی که باید راه اندازی شده و پردازش یا درخواست را اجرا کنند برای کاربر نمایش داده و به دنبال آن کاربر می تواند تصمیم بگیرد کدام کامپوننت باید اجرا شود.

### 2-3-2- تعریف intent filter

همان طور که گفته شد سیستم اندروید به وسیله ی intent filter تشخیص می دهد آیا یک برنامه ی کاربردی قابلیت احرای درخواست معینی را دارد یا خیر. در سیستم اندروید برنامه نویس می تواند با استفاده از intent filter امکاناتی که نرم افزار دارد را اعلان کند.

می توانید کامپوننت های اندرویدی خود را به وسیله ی intent filter برای event و رخدادهای خاص معرفی کنید (در واقع به کامپوننت ها اعلان کنید که به رخدادهای مورد نظر گوش داده و منتظر فعال شدن آن ها باشند). در صورتی که این کار برای کامپوننت معینی انجام نشده باشد، آنگاه فقط intent های صریح (explicit) می توانند آن کامپوننت را فراخوانی کنند. مبحث حاضر نحوه ی ثبت یک کامپوننت برای intent معین را با ذکر مثال شرح می دهد. نکته ی مهم و کلیدی این است که کامپوننت ها برای action, mime-type مناسب ثبت نام کرده و گوش فرادهند و meta-data (اطلاعاتی پایه ای درباره ی intent) مربوطه را دربر گیرد.

اگر شما چنین intent ای را به سیستم اندروید ارسال کنید، سیستم تمامی کامپوننت هایی که برای این intent به ثبت رسیده اند را شناسایی می کند. در صورتی که چندین کامپوننت برای این intent ثبت شده باشند، سیستم اندروید آن ها را برای کاربر لیست می کند و کاربر می تواند تصمیم بگیرد با کامپوننت درخواست اجرا شود.

### مثال: ثبت و تخصیص یک activity به عنوان مرورگر

کد زیر یک activity را برای intent ثبت می کند. این زمانی که کاربر می خواهد یک صفحه وب را باز کند، فعال شده و فراخوانی می گردد.

```
<activity android:name=".BrowserActivity"
    android:label="@string/app_name">
<intent-filter>
<action android:name="android.intent.action.VIEW" />
<category android:name="android.intent.category.DEFAULT" />
<data android:scheme="http"/>
</intent-filter>
</activity>
```

## مثال: ثبت و تخصیص یک activity برای اشتراک گذاری و ارسال اطلاعات

مثال زیر یک activity ویژه‌ی ACTION\_SEND ثبت می کند. همان طور که در کد زیر می بینید حاضر تنها قادر به پردازش داده‌هایی از نوع text/plain می باشد (با استفاده از intent-filter خود را مربوط به این داده اعلان یا ثبت کرده است).

```
<activity
    android:name=".ActivityTest"
    android:label="@string/app_name" >
<intent-filter>
<action android:name="android.intent.action.SEND" />
<category android:name="android.intent.category.DEFAULT" />
<data android:mimeType="text/plain" />
</intent-filter>
</activity>
```

اگر یک کامپوننت intent filter تعریف نکند، در آن صورت فقط intent صریح می تواند آن را فراخوانی کند.

## 4-2-گوش فرا دادن به event ها (ارسال پیغام به سیستم اندروید از طریق intent)

با استفاده از intent ها می توان پیغام هایی (broadcast message) را به سیستم اندروید ارسال کرد. Broadcast receiver می تواند به رخدادی گوش فرا داده (register کرده) و زمانی که رخداد اتفاق افتاد، از آن مطلع شود.

اپلیکیشن شما نیز می تواند به رخدادهای سیستم همچون دریافت ایمیل جدید، بالا آمدن سیستم، تماس جدید گوش فرا داده، و متناسب با آن عکس العمل مربوطه را نشان دهد.

## ۵-۲-شناختی intent receiver های مربوطه/بررسی اینکه آیا یک کامپوننت به intent خاصی گوش فرا می دهد/برای آن ثبت شده یا خیر

گاهی لازم می شود برنامه نویس بررسی کرده و مطمئن شوید آیا کامپوننت خاص برای یک intent ثبت شده و به آن گوش می دهد یا خیر. برای مثال، می خواهید ببینید آیا intent receiver خاصی (یک کامپوننت) وجود دارد یا خیر، و در صورت وجود آن کامپوننت قابلیت خاصی را در اپلیکیشن خود فعال نمایید.

برای این منظور می توانید از کلاس PackageManager استفاده کنید.

نمونه کد زیر بررسی می کند آیا کامپوننتی برای intent مورد نظر ثبت شده است یا خیر. کافی است intent خود را ساخته و آن را به متد زیر به عنوان آرگومان ارسال کنید.

```
public static boolean isIntentAvailable(Context ctx, Intent intent) {
    final PackageManager mgr = ctx.getPackageManager();
    List<ResolveInfo> list =
        mgr.queryIntentActivities(intent,
            PackageManager.MATCH_DEFAULT_ONLY);
    return list.size() > 0;
}
```

با توجه به نتیجه ی بازگشتی می توانید اپلیکیشن خود را تنظیم کنید. برای مثال، می توانید برخی از آیتم های منو را غیرفعال ساخته یا پنهان نمایید.

## تمرین: راه اندازی activity به وسیله intent صریح

تمرین زیر نشان می دهد چگونه با استفاده از یک intent صریح، یک subactivity را راه اندازی نموده و به آن داده هایی را ارسال کنید.

## ۵-۲-۱- ساخت پروژه و فایل layout اصلی

یک پروژه جدید اندروید به نام com.vogella.android.intent.explicit ایجاد کرده (این اسم را در فیلد package وارد نمایید) و activity آن را MainActivity نام گذاری نمایید.

حال محتويات فایل layout اين activity را به صورت زير ويرايش نمایيد:

```
<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"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >
    <EditText
        android:id="@+id/inputforintent"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:minHeight="60dip"
        android:text="First Activity"
        android:textSize="20sp" >
    </EditText>
    <Button
        android:id="@+id/startintent"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/inputforintent"
        android:layout_below="@+id/inputforintent"
        android:onClick="onClick"
        android:text="Calling an intent" />
</RelativeLayout>
```

## ۵-۲-۲- ایجاد یک فایل layout جدید

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <TextView
        android:id="@+id/displayintentextra"
        android:layout_width="wrap_content"
```

```

    android:layout_height="wrap_content"
    android:text="Input"
/>
<EditText
    android:id="@+id/returnValue"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >
    <requestFocus />
</EditText>
</LinearLayout>

```

### ۳-۵-۲- ایجاد یک activity دیگر

کلاس activity دیگری به نام ResultActivity تعریف نموده و کد موجود در بدنهٔ این کلاس را مانند نمونهٔ زیر نگارش نمایید.

```

package com.vogella.android.intent.explicit;
import android.app.Activity;
import android.os.Bundle;
public class ResultActivity extends Activity {
    @Override
    public void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        setContentView(R.layout.activity_result);
    }
}

```

کلاس فوق را به فایل AndroidManifest.xml اضافه نمایید.

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.vogella.android.first"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk android:minSdkVersion="14" />
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:label="@string/app_name"
            android:name=".MainActivity" >
            <intent-filter >
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:label="Result Activity"
            android:name=".ResultActivity" >
        </activity>
    
```

```
</application>
</manifest>
```

**نکته:** از آنجایی که activity ذکر شده توسط acitivity دیگری (کلاس MainActivity) راه اندازی می شود، از این به بعد تحت عنوان subactivity به آن اشاره می کنیم.

## 4-5-2- راه اندازی subactivity

MainActivity را با کلیک بر روی یک دکمه در صفحه ای اصلی برنامه که همان کلاس Subactivity محسوب می شود، راه اندازی نمایید. کد زیر شما را در خصوص پیاده سازی این عملیات راهنمایی می کند. با نوشتن دستورات مناسب بجای TODO ها در کد برنامه، اپلیکیشن خود را طوری بنویسید که ResultActivity که فراخوانی متده OnClick() راه اندازی شود (به محض کلیک بر روی دکمه در صفحه ای اصلی، اپلیکیشن کاربر را به activity دوم هدایت کند).

```
package com.vogella.android.intent.explicit;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
public class MainActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void onClick(View view) {
        EditText text = (EditText) findViewById(R.id.inputforintent);
        // used later
        String value = text.getText().toString();
        // TODO 1 create new Intent(context, class)
        // use the activity as context parameter
        // and "ResultActivity.class" for the class parameter
        // TODO 2 start second activity with
        // startActivityForResult(intent);
    }
}
```

پس از اتمام این بخش از تمرین، اپلیکیشن خود را راه اندازی نموده و مطمئن شوید که کاربر با کلیک بر روی دکمه در صفحه ای اول اپلیکیشن، به activity دوم راه پیدا می کند.

## ۵-۵-۲- انتقال داده (ارسال مقدار) از activity اول به activity دوم (ResultActivity)

کلاس MainActivity می بایست مقدار EditText را به subactivity ارسال کند. برای این منظور کافی است متدهای putExtra("yourKey", string) را بر روی آبجکت Intent بکار ببرید.

## ۵-۵-۳- دریافت اطلاعات ارسالی از activity اول (داده های intent) در ResultActivity

حال داده های ارسالی از activity اصلی اپلیکیشن را از Bundle که بخشی از آبجکت intent است با فراخوانی متدهای getIntent().getExtras() در کلاس ResultActivity بازیابی نمایید.

حال مقدار extra را با فراخوانی متدهای extras.getString("yourkey") بر روی آبجکت bundle که در نتیجه ای فراخوانی متدهای getExtras() بدست آوردید، بازیابی نمایید.

این مقدار می بایست داخل کلاس TextView با ID یا شناسه ای displayIntentExtra جایگذاری شود.

## تمرین: بازیابی داده ها از ResultActivity (انتقال داده ها از activity دوم به activity اصلی برنامه)

در تمرین زیر کدی به متن برنامه اضافه می کنید که با کلیک کاربر بر روی دکمه Back در activity دوم، داده هایی را از subactivity اصلی اپلیکیشن ارسال کند.

برای نیل به این هدف، متدهای finish() در کلاس ResultActivity پیاده سازی می کنید.

```
@Override
public void finish() {
    // TODO 1 create new Intent
    // Intent intent = new Intent();
    // TODO 2 read the data of the EditText field
    // with the id returnValue
    // TODO 3 put the text from EditText
    // as String extra into the intent
    // use editText.getText().toString();
    // TODO 4 use setResult(RESULT_OK, intent);
    // to return the Intent to the application
```

```

    super.finish();
}
}

```

در جایگاه TODO ها در کد فوق، بدنہ ی متده finish() را پیاده سازی نمایید.

Solve all TODOs.

## 7-5-2- دریافت و ارزیابی داده های بازگشته (ارسالی از activity دوم) در

### کلاس MainActivity

متده startActivityForResult() را در کلاس MainActivity فراخوانی نمایید تا راه اندازی شود. این کار به شما امکان می دهد تا متده onActivityResult() را برای دریافت داده از activity دوم فراخوانی نمایید. حال داده های ارسالی extra که همراه با آبجکت bundle از activity دوم ارسال شده، استخراج نمایید.

بد نیست همراه داده های extra یک پیغام Toast نیز برای کاربر نمایش داده و اطمینان دهید که داده ها طبق انتظار دریافت شدند. کد زیر شما را در خصوص پیاده سازی این عملیات راهنمایی می کند:

```
package com.vogella.android.intent.explicit;
```

```

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
public class MainActivity extends Activity {
    // constant to determine which sub-activity returns
    private static final int REQUEST_CODE = 10;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void onClick(View view) {
        EditText text = (EditText) findViewById(R.id.inputforintent);
        String string = text.getText().toString();
        Intent i = new Intent(this, ResultActivity.class);
        i.putExtra("yourkey", string);
        // TODO 2.. now use
        // startActivityForResult(i, REQUEST_CODE);
    }
}

```

```

// TODO 3 Implement this method
// assumes that "returnkey" is used as key to return the result
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (resultCode == RESULT_OK && requestCode == REQUEST_CODE) {
        if (data.hasExtra("returnkey")) {
            String result = data.getExtras().getString("returnkey");
            if (result != null && result.length() > 0) {
                Toast.makeText(this, result, Toast.LENGTH_SHORT).show();
            }
        }
    }
}

```

## راه حل: استفاده از intent ها

با تکمیل این تمرین، پیاده سازی activity شما می بایست مشابه کد کلاس های زیر باشد.

```

package com.vogella.android.intent.explicit;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.widget.EditText;
import android.widget.TextView;
public class ResultActivity extends Activity {
    @Override
    public void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        setContentView(R.layout.activity_result);
        Bundle extras = getIntent().getExtras();
        String inputString = extras.getString("yourkey");
        TextView view = (TextView) findViewById(R.id.displayintentextra);
        view.setText(inputString);
    }
    @Override
    public void finish() {
        Intent intent = new Intent();
        EditText editText= (EditText) findViewById(R.id.returnValue);
        String string = editText.getText().toString();
        intent.putExtra("returnkey", string);
        setResult(RESULT_OK, intent);
        super.finish();
    }
}

```

```

package com.vogella.android.intent.explicit;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends Activity {
    // constant to determine which sub-activity returns
    private static final int REQUEST_CODE = 10;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void onClick(View view) {
        EditText text = (EditText) findViewById(R.id.inputforintent);
        String string = text.getText().toString();
        Intent i = new Intent(this, ResultActivity.class);
        i.putExtra("yourkey", string);
        startActivityForResult(i, REQUEST_CODE);
    }
    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (resultCode == RESULT_OK && requestCode == REQUEST_CODE) {
            if (data.getStringExtra("returnkey")) {
                String result = data.getStringExtra("returnkey");
                if (result != null && result.length() > 0) {
                    Toast.makeText(this, result, Toast.LENGTH_SHORT).show();
                }
            }
        }
    }
}

```

## تمرین: استفاده از intent به اشتراک گذاری و تبادل اطلاعات

یک دکمه‌ی دیگر به فایل layout مربوط به activity خود اضافه نمایید. کلیک بر روی این دکمه می‌بایست این امکان را فراهم بیاورد تا متن EditText را به اشتراک بگذارید. برای ارسال اطلاعات از اپلیکیشن دیگر، می‌بایست مقدار مربوطه را در ثابت Intent.ACTION\_SEND قرار دهید. سپس یک متده با این تعریف public void shareData(View view) بر اساس تکه کد زیر در activity خود ایجاد نمایید.

```
Intent sharingIntent = new Intent(Intent.ACTION_SEND);
sharingIntent.setType("text/plain");
sharingIntent.putExtra(android.content.Intent.EXTRA_TEXT, "REPLACE WITH YOUR TEXT");
startActivity(sharingIntent);
```

این متده را به وسیله `i` property یا خصوصیت `android:onClick` در فایل `layout` خود به دکمه `i` مربوطه متصل نمایید. سپس این برنامه را اجرا نموده و بر روی دکمه `i` مورد نظر کلیک کنید. در محیط شبیه ساز، قاعده تناها برنامه ای که به این intent گوش فرا می دهد و در سیستم اندروید برای آن ثبت شده و متعاقباً مستقیماً راه اندازی می شود، اپلیکیشن ارسال SMS می باشد.

## تمرین: تخصیص و ثبت یک activity به عنوان مرورگر

### هدف اصلی

در تمرین زیر یک activity را در فایل XML خود به عنوان مرورگر ثبت می کنید. پس از این کار، هر زمان که کاربر می خواهد آدرس URL ای که با `http` آغاز می شود را مشاهده کند، activity ای ذکر شده، برای پردازش intent مربوطه در دسترس خواهد بود.

نام بردہ کد HTML این صفحه را دانلود کرده و آن را در یک `TextView` به نمایش می گذارد.

### ۵-۸-۲- ایجاد پروژه

یک پروژه را در Android Studio ایجاد کنید و `activity_main.xml` را از `res/layout` حذف کنید. در جای آن، فایل `BrowserActivity.java` را ایجاد کنید و آن را در `src/main/java` قرار دهید.

### ۵-۹-۲- ثبت و تخصیص یک activity به عنوان مرورگر

در فایل `AndroidManifest.xml`، `activity` را در `manifest` بخش اضافه کنید. این `activity` را با نام `BrowserActivity` و `de.vogella.android.intent.browserfilter` معرفی کنید. همچنین `intent-filter` را اضافه کنید که `action` مقدار داشته باشد `Intent.Action_VIEW` و `category` مقدار داشته باشد `Intent.Category_BROWSING`.

اجازه‌ی دسترسی به اینترنت را از طریق تگ uses-permission لازم است در فایل manifest تنظیم نمایید.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="de.vogella.android.intent.browserfilter"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk android:minSdkVersion="15" />
    <uses-permission android:name="android.permission.INTERNET" >
    </uses-permission>
    <application
        android:icon="@drawable/icon"
        android:label="@string/app_name" >
        <activity
            android:name=".BrowserActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.VIEW" />
                <category android:name="android.intent.category.DEFAULT" />
                <data android:scheme="http" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

حال فایل layout مربوطه را به صورت زیر ویرایش نمایید.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    >
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/textView"/>
</LinearLayout>
```

کد کلاس activity را نیز به ترتیب زیر ویرایش نمایید.

```
package de.vogella.android.intent.browserfilter;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.URL;
import android.app.Activity;
import android.content.Intent;
```

```

import android.net.Uri;
import android.os.Bundle;
import android.os.StrictMode;
import android.widget.TextView;
public class BrowserActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // To keep this example simple, we allow network access
        // in the user interface thread
        StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder()
            .permitAll().build();
        StrictMode.setThreadPolicy(policy);
        setContentView(R.layout.main);
        Intent intent = getIntent();
        TextView text = (TextView) findViewById(R.id.textView);
        // To get the action of the intent use
        String action = intent.getAction();
        if (!action.equals(Intent.ACTION_VIEW)) {
            throw new RuntimeException("Should not happen");
        }
        // To get the data use
        Uri data = intent.getData();
        URL url;
        try {
            url = new URL(data.getScheme(), data.getHost(), data.getPath());
            BufferedReader rd = new BufferedReader(new InputStreamReader(
                url.openStream()));
            String line = "";
            while ((line = rd.readLine()) != null) {
                text.append(line);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

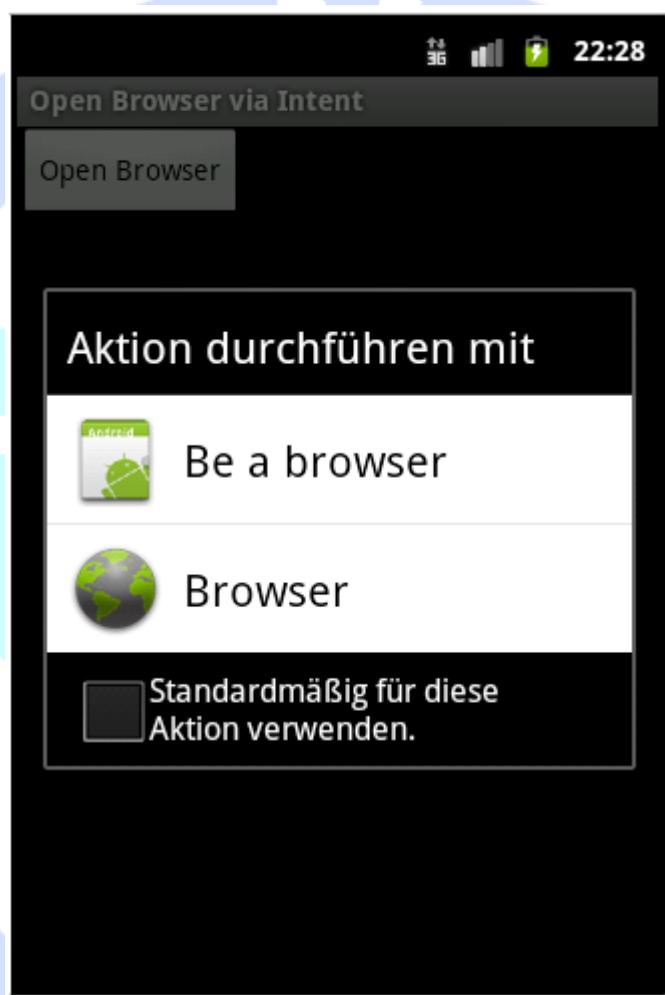
## ۲-۵-۲- تست برنامه

اپلیکیشن خود را نصب نمایید. زمانی که یک intent را جهت مشاهده ی محتویات صفحه ی URL فراخوانی یا فعال می کنید، کاربر قاعدها باید بتواند پیاده سازی سفارشی مرورگر شما را انتخاب کند.

در این راستا یک دکمه ی اضافی به اپلیکیشن خود اضافه می کنید که با کلیک بر روی آن کد فراخوانی می شود.

```
intent = new Intent(Intent.ACTION_VIEW,Uri.parse("http://www.vogella.com"));
startActivity(intent);
```

پس از انتخاب کامپوننت دلخواه از میان کامپوننت های پیشنهادی توسط سیستم، کد HTML مورد نظر ابتدا در حافظه بارگذاری شده، سپس در المان رابط کاربری TextView به نمایش در می آید.





## ۵-۲- تبدیل اپلیکیشن به یک مرورگر واقعی

در صورت تمایل می توانید TextView را با یک WebView جایگزین نموده و اپلیکیشن خود را به یک مرورگر واقعی تبدیل نمایید. WebView خود درخواست HTTP را برای شما مدیریت و بارگذاری می کند. کافی است آدرس URL را به واسطه ای متده loadURL به آن تخصیص دهید.

## تمرین: انتخاب یک عکس به وسیله ای آبجکت intent

## هدف

مثال زیر نمایش می دهد چگونه می توان با استفاده از یک intent، عکس دلخواه را از اپلیکیشن های تخصیص یافته برای این منظور در سیستم اندروید، انتخاب نمود.

### ایجاد پروژه

یک پروژه ای جدید اندروید و یک activity به ترتیب با نام های activity و de.vogella.android.imagepick ایجاد نمایید.

فایل layout مربوط به صورت زیر ویرایش نمایید.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="pickImage"
        android:text="Button" >
    </Button>
    <ImageView
        android:id="@+id/result"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:src="@drawable/icon" >
    </ImageView>
</LinearLayout>
```

کد کلاس activity خود را به صورت زیر ویرایش نمایید.

```
package de.vogella.android.imagepick;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;

import android.app.Activity;
import android.content.Intent;
import android.graphics.Bitmap;
```

```

import android.graphics.BitmapFactory;
import android.os.Bundle;
import android.view.View;
import android.widget.ImageView;
public class ImagePickActivity extends Activity {
    private static final int REQUEST_CODE = 1;
    private Bitmap bitmap;
    private ImageView imageView;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        imageView = (ImageView) findViewById(R.id.result);
    }
    public void pickImage(View View) {
        Intent intent = new Intent();
        intent.setType("image/*");
        intent.setAction(Intent.ACTION_GET_CONTENT);
        intent.addCategory(Intent.CATEGORY_OPENABLE);
        startActivityForResult(intent, REQUEST_CODE);
    }
    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        InputStream stream = null;
        if (requestCode == REQUEST_CODE && resultCode == Activity.RESULT_OK)
            try {
                // recycle unused bitmaps
                if (bitmap != null) {
                    bitmap.recycle();
                }
                stream = getContentResolver().openInputStream(data.getData());
                bitmap = BitmapFactory.decodeStream(stream);
                imageView.setImageBitmap(bitmap);
            } catch (FileNotFoundException e) {
                e.printStackTrace();
            } finally {
                if (stream != null)
                    try {
                        stream.close();
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
            }
    }
}

```

تست اپلیکیشن

با توجه به کدی که برای اپلیکیشن خود نوشته اید، باید پس از اجرای آن بتوانید عکس دلخواه را از ImageView مستقر در دستگاه اندروید خود انتخاب نموده و سپس آن را به تخصیص دهید.

### تمرین: استفاده از intent های ضمنی جهت راه اندازی activity ها

هدف

تمرین زیر نحوه‌ی استفاده از intent های ضمنی جهت راه اندازی activity ها در سیستم اندروید را برای شما شرح می‌دهد.

ایجاد پروژه

یک پروژه‌ی جدید و یک activity به ترتیب به نام های de.vogella.android.intent.implicit و CallIntentsActivity ایجاد نمایید.

در این مثال، با پیاده‌سازی کلاس Spinner، یک لیست کشویی ایجاد می‌کنید که به کاربر اجازه می‌دهد تا خود تصمیم بگیرد کدام intent را فراخوانی کند. محتوای این لیست کشویی (Spinner) را مقادیر static و ثابت تعریف خواهید نمود.

برای این منظور فایل intents.xml زیر را در پوشه‌ی res/values ایجاد نمایید.

```
<resources>
    <string-array name="intents">
        <item>Open Browser</item>
        <item>Call Someone</item>
        <item>Dial</item>
        <item>Show Map</item>
        <item>Search on Map</item>
        <item>Take picture</item>
        <item>Show contacts</item>
        <item>Edit first contact</item>
    </string-array>
</resources>
```

فایل layout مربوط به activity مورد نظر را به صورت زیر ویرایش نمایید.

Change the layout file of the activity to the following.

```
<?xml version="1.0" encoding="utf-8"?>
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:alignmentMode="alignBounds"
    android:columnCount="1" >
    <Spinner
        android:id="@+id/spinner"
        android:layout_gravity="fill_horizontal"
        android:drawSelectorOnTop="true"
        >
    </Spinner>
    <Button
        android:id="@+id/trigger"
        android:onClick="onClick"
        android:text="Trigger Intent" >
    </Button>
</GridLayout>
```

برای اینکه بتوانید از intent های مورد نظر استفاده کنید، لازم است مجوزهای مربوطه را در فایل تنظیمات AndroidManifest.xml ثبت نمایید. در واقع فایل تنظیمات اپلیکیشن شما می باشد.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="de.vogella.android.intent.implicit"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk android:minSdkVersion="15" />
    <uses-permission android:name="android.permission.CALL_PRIVILEGED" >
    </uses-permission>
    <uses-permission android:name="android.permission.CALL_PHONE" >
    </uses-permission>
    <uses-permission android:name="android.permission.CAMERA" >
    </uses-permission>
    <uses-permission android:name="android.permission.READ_CONTACTS" >
    </uses-permission>
    <uses-permission android:name="android.permission.INTERNET"/>
    <application
        android:icon="@drawable/icon"
        android:label="@string/app_name" >
        <activity
            android:name=".CallIntentsActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

```
</activity>
</application>
</manifest>
```

کد کلاس activity خود را به صورت زیر ویرایش نمایید.

Change your activity class to the following code.

```
package de.vogella.android.intent.implicit;
import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.Spinner;
import android.widget.Toast;
public class CallIntentsActivity extends Activity {
    private Spinner spinner;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        spinner = (Spinner) findViewById(R.id.spinner);
        ArrayAdapter adapter = ArrayAdapter.createFromResource(this,
                R.array.intents, android.R.layout.simple_spinner_item);
        adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        spinner.setAdapter(adapter);
    }
    public void onClick(View view) {
        int position = spinner.getSelectedItemPosition();
        Intent intent = null;
        switch (position) {
            case 0:
                intent = new Intent(Intent.ACTION_VIEW,
                        Uri.parse("http://www.vogella.com"));
                break;
            case 1:
                intent = new Intent(Intent.ACTION_CALL,
                        Uri.parse("tel:(+49)12345789"));
                break;
            case 2:
                intent = new Intent(Intent.ACTION_DIAL,
                        Uri.parse("tel:(+49)12345789"));
                startActivity(intent);
                break;
            case 3:
                intent = new Intent(Intent.ACTION_VIEW,
                        Uri.parse("geo:50.123,7.1434?z=19"));
                break;
        }
    }
}
```

```

case 4:
    intent = new Intent(Intent.ACTION_VIEW,
        Uri.parse("geo:0,0?q=query"));
    break;
case 5:
    intent = new Intent("android.media.action.IMAGE_CAPTURE");
    break;
case 6:
    intent = new Intent(Intent.ACTION_VIEW,
        Uri.parse("content://contacts/people/"));
    break;
case 7:
    intent = new Intent(Intent.ACTION_EDIT,
        Uri.parse("content://contacts/people/1"));
    break;

}
if (intent != null) {
    startActivity(intent);
}
}

@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (resultCode == Activity.RESULT_OK && requestCode == 0) {
        String result = data.toURI();
        Toast.makeText(this, result, Toast.LENGTH_LONG);
    }
}
}

```

## تست اپلیکیشن

با اجرای اپلیکیشن می بینید که یک لیست کشیوی به برنامه اضافه شده که کلیک بر روی هر یک از آن ها به اجرا و راه اندازی activity های مربوطه متنه می شود.

**توجه:** اگر بخارط داشته باشد، شما هیچ اپلیکیشن مشخصی را به عنوان دریافت کننده `intent` به طور صریح مشخص نکردید (تنها عملیاتی که باید انجام شود را قید نمودید). این امر به شما اجازه می دهد تا task هایی با وابستگی کم (loosely coupled) تعریف نمایید که قابلیت استفاده از کامپوننت های اپلیکیشن های مختلف را فراهم می آورد (در این سناریو، سیستم اندروید تمامی کامپوننت هایی که برای این `intent` تخصیص یافته و قابلیت اجرای درخواست آن را دارند، شناسایی کرده و لیستی از کامپوننت های مذبور را برای کاربر لیست می کند. سپس کاربر تصمیم می گیرد را با کدام کامپوننت باز کند).

## 2-6- مدیریت چرخه‌ی حیات / lifecycle اپلیکیشن

مدیریت چرخه‌ی حیات/**activity** اپلیکیشن و **life cycle** در اندروید  
این مبحث به شرح **life cycle** یا چرخه‌ی حیات اپلیکیشن و **activity** در اندروید می‌پردازد.

در شرایط ایده‌ال، کلیه‌ی اپلیکیشن‌هایی که کاربر در دستگاه اندروید خود راه اندازی می‌کند، باید در حافظه باقی بمانند. ناگفته‌ی پیدا است که این امر در تسريع راه اندازی مجدد اپلیکیشن بسیار موثر است. اما حقیقت امر این است که حافظه‌ی دستگاه اندروید همانند اندازه‌ی خود دستگاه، کوچک بوده و دارای ظرفیت محدود می‌باشد. به منظور مدیریت بهینه‌ی این منابع محدود سیستم اندروید این اجازه را دارد که فرایندهای در حال اجرا را متوقف کرده یا کامپوننت‌های اندروید را بازیابی (recycle) کند.

در صورتی که سیستم اندروید به این منابع نیاز داشته باشد، طبیعتاً باید این منابع را آزاد کند. برای آزاد کردن این منابع از تعدادی قانون ساده پیروی می‌کند. در واقع هر فرایند در حال اجرا دارای اولویتی است که اندروید آن‌ها را بر اساس این اولویت از حافظه پاک می‌کند.

Table 1. Priorities

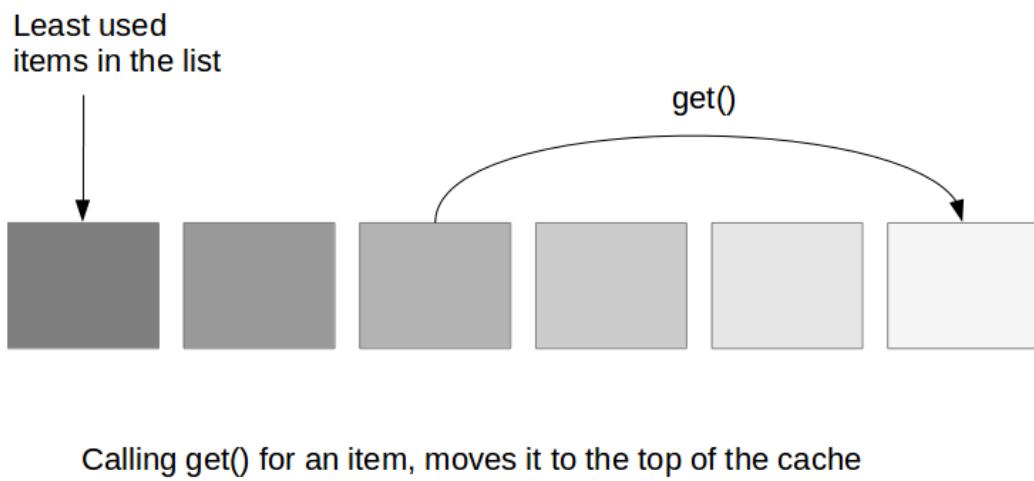
وضعیت process / status	شرح	اولویت
Foreground / پیش‌زمینه	اپلیکیشنی که کاربر در حال تعامل با آن می‌باشد یا دارای سرویسی است که به این activity متصل است. همچنین سرویسی که در حال اجرای یکی از متدهای مدیریت چرخه‌ی حیات ( <b>lifecycle method</b> ) بوده یا ای که در حال فراخوانی و اجرای متدهای <b>broadcast receiver</b> خود می‌باشد.	1

Table 1. Priorities

وضعیت process/ status	شرح	اولویت
قابل مشاهده/ Visible	کاربر دیگر با activity تعامل ندارد، اما activity هنوز (تا اندازه ای) قابل مشاهده است یا اپلیکیشن مورد نظر سرویسی دارد که توسط activity قابل مشاهده اما غیرفعال مورد استفاده قرار می گیرد.	2
Service	اپلیکیشن با یک سرویس در حال اجرا که ویژگی های 1 یا 2 را ندارد.	3
Background/ زمینه	اپلیکیشنی که تمامی activity های آن متوقف شده و هیچ سرویس یا دریافت کننده ای ندارد. سیستم اندروید تمامی آن ها را در یک لیست LRU نگه داشته و در صورت لزوم آن هایی که از تاریخ آخرین بار استفاده ای آن خیلی می گذرد یا کمترین استفاده را داشته اند را از حافظه حذف می کند.	4
Empty	اپلیکیشنی که هیچ کامپوننت در حال اجرا و فعالی ندارد.	5

تمامی فرایندهایی که از اولویت 6 برخوردار هستند (در رده ی "Empty" قرار می گیرند) به لیست LRU اضافه می شوند و فرایندهایی که در ابتدای این لیست ها قرار دارند، توسط out of memory killer نابود می شوند (برای آزاد سازی منابع از حافظه حذف می شوند). حال چنانچه اپلیکیشنی توسط کاربر مجدد راه اندازی شد، آن اپلیکیشن به انتهای صفحه هدایت می شود و اولویت آن به 1 تغییر می یابد. این جریان در تصویر زیر به نمایش گذاشته شده است:

# LRU Cache



فراخوانی متد (get) برای یک آیتم سبب می شود آن آیتم به انتهای حافظه (cache) LRU انتقال یابد.

## 7-آبجکت application

هر زمان که یک کامپوننت راه اندازی می شود، سیستم اندروید به صورت خودکار یک آبجکت application می کند که این آبجکت تحت یک فرایند جدید با ID منحصر بفرد و مختص به یک کاربر خاص ایجاد می شود. حتی اگر برنامه نویس آبجکت application را در فایل تنظیمات (AndroidManifest.xml)، در تگ Application مشخص نکند، سیستم اندروید به صورت پیش فرض یک نمونه از کلاس Application (همان آبجکت application) ایجاد می کند (این آبجکت بیشتر برای نگهداری وضعیت و اطلاعات سراسری اپلیکیشن مورد استفاده قرار می گیرد).

آبجکت ذکر شده یک سری توابع مربوط به مدیریت lifecycle ارائه می دهد که در زیر به هریک همراه با عملکرد آن اشاره می کنیم:

- **onCreate()** – این متد قبل از اینکه اولین کامپوننت های اپلیکیشن راه اندازی شوند، فراخوانی می گردد. به عبارت دیگر این متد یک تابع callback است و هنگامی که برای نخستین بار اجرا می شود، فراخوانی می گردد.

- زمانی صدا خورده می شود که کل سیستم با کمبود حافظه مواجه شده و فرایندهای در حال اجرا یا فعال می باشد میزان استفاده از خود از حافظه را کاهش دهند. سیستم اندروید با فراخوانی این تابع به اپلیکیشن دستور می دهد که باید حافظه را بهینه استفاده کرده و میزان استفاده از خود را تا حد امکان محدود کند. شما این متد را بایستی زمانی پیاده سازی کنید که لازم باشد cache یا دیگر منابع غیر ضروری که برنامه اشغال کرده را آزاد سازی نمایید.
  - این متد زمانی صدا خورده می شود که سیستم عامل احساس کند که فرایند باید میزان استفاده از خود از حافظه را کاهش دهد. این اتفاق (برای مثال) زمانی می افتد که فرایند در وضعیت background قرار گرفته باشد و در این میان حافظه ای کافی برای فعال نگه داشتن تعداد دلخواه فرایند پس زمینه ای وجود ندارد. به عبارت دیگر، این متد زمانی فراخوانده می شود که سیستم اندروید از اپلیکیشن درخواست استفاده ای بهینه از حافظه/کاهش استفاده از آن را داشته باشد. این متد پارامترهایی دارد که از وضعیت جاری اپلیکیشن خبر می دهند. برای مثال، ثابت TRIM\_MEMORY\_MODERATE بیانگر این است که فرایند در اواسط لیست LRU قرار دارد. در این شرایط طبیعتاً آزاد سازی حافظه به سیستم کمک می کند تا دیگر فرایندهای در حال اجرا را جهت بهبود کارایی کلی سیستم، در انتهای لیست مذکور نگه دارد.
  - تنها به منظور تست در محیط شبیه ساز بکار رفته و به هیچ وجه در بستر اجرا نهایی (محیط production) اپلیکیشن فراخوانی نمی شود.
  - این متد زمانی فراخوانده می شود که کامپوننت شما در حال اجرا است و در این حین تنظیمات و configuration دستگاه تغییر می کند.
- لازم به توضیح است که آیجکت application قبل از راه اندازی هر کامپوننتی ایجاد شده و حداقل تا زمانی که کامپوننت دیگری از اپلیکیشن راه اندازی نشده، به اجرا ادامه می دهد.

## content provider/چرخه ای حیات/LifeCycle-2-8

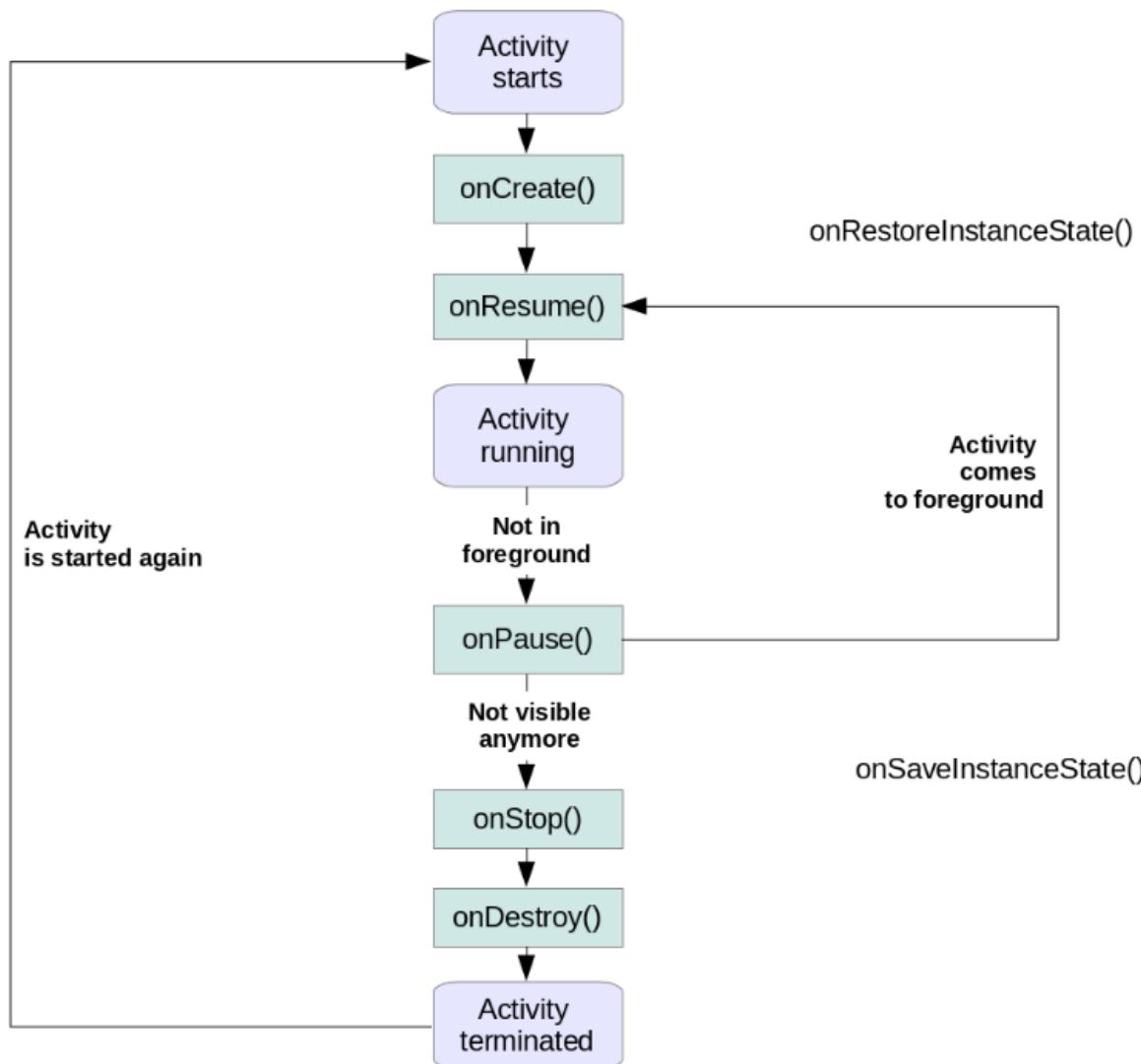
هر زمان که content provider مورد دسترسی قرار می گیرد، این کامپوننت هیچگاه به صورت انفرادی و جداگانه متوقف نمی شود. به بیانی روشنتر این کامپوننت تنها زمانی کاملاً متوقف می شود که کل فرایند اپلیکیشن خاتمه یابد.



Table 2. Activity state

وضعیت/state	شرح
Running (در حال اجرا و قابل تعامل مستقیم)	در این وضعیت، کاملا قابل مشاهده بوده و کاربر می تواند با آن مستقیما تعامل داشته باشد.
Paused (متوقف شده) اما همچنان قابل رویت	Activity is still visible but partially obscured, instance is running but might be killed by the system. در این وضعیت، Activity همچنان قابل مشاهده است ، اما نه به طور کامل و به اصطلاح دیگر در foreground قرار ندارد. در واقع نمونه در حال اجرا است، اما ممکن است سیستم اندروید آن را از حافظه پاک کند .(kill)
Stopped (متوقف شده و غیر قابل رویت)	Activity is not visible, instance is running but might be killed by the system. قابل مشاهده نیست، نمونه در حال اجرا است اما ممکن است توسط سیستم از حافظه پاک شود.
Killed (کاملا نابود یا به اصطلاح از حافظه پاک شده است)	Activity has been terminated by the system of by a call to its finish() method. در این وضعیت سیستم اندروید متد finish() را صدا زده و activity مورد نظر در پی اجرای شدن این متد از حافظه پاک شده است.

در نمودار زیر چرخه‌ی حیات یک activity را به همراه متدهای مهم آن مشاهده می‌کنید.



لازم به ذکر است که اندروید متدهای دیگری نیز جهت چرخه‌ی حیات در اختیار دارد، اما موارد استفاده از آن‌ها نادر است. به عنوان مثال می‌توان به متد `onDestroy()` اشاره کرد.

## ۹-۲-۲-پایان یافتن activity ها توسط سیستم عامل

سیستم اندروید این اجازه را دارد که در راستای بهبود کارایی و بهینه سازی سرعت، با recycle کردن activity ها (استفاده ای مجدد)، منابعی را آزاد کند. گفته می شود که سیستم عامل اندروید می تواند activity ها را فارغ از کل فرایند اپلیکیشن (به صورت منفرد) خاتمه دهد، در حالی که واقعیت چیز دیگری است. سیستم اندروید هیچگاه activity های فردی را recycle نمی کند، بلکه کل فرایند را بر اساس قوانین مشخص پایان می دهد.

**توجه:** این یک باور غلط است که اندروید activity های منفرد را از حافظه پاک می کند. متاسفانه این توضیح اشتباه در سایت رسمی توسعه اندروید نیز ذکر شده است. اما بنا به گفته ای Dianne Hackborn که عضو تیم طراحان و توسعه دهندگان اندروید، بخش پیاده سازی out of memory killer هست، اندروید هیچگاه activity ها را به صورت منفرد خاتمه نمی دهد، بلکه کل فرایند میزبان activity ها را پایان می دهد.

**توجه:** یک برای اینکه اطلاعات و وضعیت خود را به درستی ذخیره کرده و بعده ها بتواند آن را بازیابی کند، ناگذیر است اطلاعات state را در زمان مناسب ذخیره کند. همچنین، در صورتی که activity های غیر ضروری دیگر قابل مشاهده یا دسترسی نبودند، می بایست عملیات غیر ضروری را به منظور صرفه جویی در منابع سیستم متوقف کند.

برای متوقف کردن listener های framework (گوش فرخوان ها) و بروز رسانی های UI، می بایست متد onPause() را فراخوانی کرده و سپس با استفاده از متد onStop() داده های اپلیکیشن را ذخیره نمایید. دوتابع نام بردۀ حتما پیش از خاتمه یافتن activity صدا خورده می شوند. با فراخوانی onResume() می توانید دوباره listener ها را تخصیص داده و در صورت لزوم، بروز رسانی های UI را بر اساس داده های ذخیره شده از سر بگیرید (مجددا راه اندازی نمایید).

علاوه بر مدیریت منابع (resource management)، اندروید می بایست activity ها را در صورت رخداد تغییر در تنظیمات دستگاه، از نو ایجاد کند. آجکت Configuration دربردارنده ی تنظیمات و

بیکری‌نگاری دستگاه می‌باشد. اگر این تنظیمات تغییر کند، activity های مجدد راه اندازی می‌شوند چرا که لازم است برای این تنظیمات جدید از منابع/resource متفاوت استفاده کنند.

## 10-2-داده ها و اطلاعات ذخیره شده از آبجکت activity جهت بازگردانی activity instance state قبلی

همان طور که در بالا ذکر شد، هنگامی تنظیمات دستگاه تغییر می‌کند، مثلاً وضعیت و جهت نمایش تغییر می‌یابد، activity برای اینکه متناسب با تنظیمات جدید نمایش داده شود، باید مقادیر و منابع متفاوتی را بخواند و از اینرو از نو ساخته می‌شود (ابتدا متد() onDestroy() صدا خورده و بلافاصله متد() onCreate() فراخوانی می‌شود). حال به منظور اینکه سیستم اندروید بتواند activity را با تنظیمات جدید، اما وضعیت قبلی بازگردانی کند، ناچار نمونه‌ای از آن کلاس را ذخیره می‌کند. سپس بر اساس داده‌های ذخیره شده، activity را به حالت قبلی بازمی‌گرداند. Instance state در واقع داده‌های موقتی و غیر ماندگار هستند که باید در بازه‌ی زمانی تغییر بین دو وضعیت نمایش (برای مثال بخارط تغییر در تنظیمات دستگاه) بین activity‌ها (نمونه‌ی ذخیره شده از activity و نمونه‌ی مجدد ایجاد شده) پاس داده شود تا اطلاعات و انتخاب‌های کاربر (اینک با تنظیمات جدید) به حالت قبل بازگردانی شود. این اپلیکیشن است که وظیفه‌ی بازگردانی خود به حالت قبلی را دارد.

Instance state معمولاً جفت‌های کلید-مقدار هستند که از activity خاتمه یافته، در آبجکت Bundle ذخیره می‌شوند و جهت بازگردانی وضعیت activity به حالت قبلی (وضعیتی که کاربر در آن اپلیکیشن را ترک کرد) مورد استفاده قرار می‌گیرند.

سناریویی را در نظر بگیرید که کاربر با نوار پیمایش به پایین یک لیست می‌رود (ListView) که در آن هزاران آیتم وجود دارد و در این میان activity حذف و مجدد ساخته می‌شود. طبیعتاً کاربر دوست ندارد موقعیت جاری خود را در این لیست بسیار بزرگ از دست بدهد و دوباره مجبور به پیدا کردن آن شود. از اینرو موقعیت جاری کاربر باید حفظ شده و پس از ایجاد activity دوباره کاربر به آن بازگردانده شود.

برای این منظور متد (`onSaveInstanceState()`) صدا خورده شده و وضعیت نمونه را در قالب یک آبجکت `Bundle` در خود ذخیره می کند. آبجکت `Bundle` می تواند انواع داده هی اولیه، آرایه، رشته/`String` و `Object` هایی از جنس `Serializable` و `Parcelable` را در خود ذخیره کند.

داده های ذخیره شده در کلاس `Bundle`، در زمان راه اندازی مجدد `activity` مورد نظر به عنوان پارامتر به متدهای `onCreate()` و `onRestoreInstanceState()` ارسال می شوند.

در صورت بازنویسی `(override)` `onSaveInstanceState()` پیاده سازی توابع `onRestoreInstanceState()` لازم است پیاده سازی کلاس پدر (`super implementation`) را نیز فراخوانی نمایید زیرا `view` های پیش فرض اندروید داده های خود را به واسطه هی فراخوانی `onSaveInstanceState()` از متد `View.onSaveInstanceState()` محتوا خود را از طریق فراخوانی پیش فرض این متد ذخیره می کند.

با استفاده از دو تابع `onCreate()` و `onRestoreInstanceState()` می توانید `activity` را (در صورتی که `activity` نابود شده و مجدداً ساخته شد) بازسازی نمایید.

**توجه:** برای بازگردانی وضعیت `activity` (بازیابی `instance state`) توصیه می کنیم از متد `onRestoreInstanceState()` استفاده نمایید. این روش تنظیم و راه اندازی اولیه `activity` را از بازگردانی وضعیت آن به حالت قبلی (`instance state`) جدا می سازد.

اگر کاربر با `activity` تعامل داشته و سپس دکمه `Back` را فشار دهد یا اینکه متد `finish()` آن صدا خورده شود، `activity` مورد نظر از پشته هی `activity` ها حذف شده و `activity` می شود. در چنین شرایطی، هیچ `instance state` (داده هی ذخیره شده ای از وضعیت `activity` وجود ندارد که بخواهد بازگردانی شود) برای بازگردانی وجود ندارد، از اینرو متد `onSaveInstanceState()` هیچگاه فراخوانی نمی شود.

حال چنانچه کاربر با `activity` تعامل داشته و بعد دکمه `Home` را فشار دهد، اطلاعات مربوط به وضعیت `activity instance / activity` مربوط به `activity` می باشد.

`onSaveInstanceState()` فراخوانده می شود و اطلاعات مربوط به وضعیت `activity` را در خود ذخیره می کند. اکنون اگر کاربر به اپلیکیشن بازگردد یا به عبارتی آن را مجدد راه اندازی کند، سیستم عامل اندروید آخرین `activity` که هنگام ترک اپلیکیشن فعال و در حال اجرا بود را فراخوانی می کند. در واقع زمانی که `activity` مجدد راه اندازی می شود، آبجکت `Bundle` (حاصل داده های ذخیره شده از وضعیت `activity`) در اختیار متدهای `onCreate()` و `onRestoreInstanceState()` قرار می گیرد.

**توجه:** در صورتی که کاربر دکمه `Back` را فشار دهد، متدهای `onSaveInstanceState()` و `onRestoreInstanceState()` صدا زده نمی شود. توجه داشته باشید که از این روش نباید به هیچ وجه برای ذخیره داده هایی که می بایست ماندگار شوند، استفاده نمایید.

## 11-2-آبجکت های جاوایی که در صورت تغییر در تنظیمات دستگاه باید بین نمونه های `activity` پاس داده شوند

در صورتی که تنظیمات دستگاه (`configuration`) حین اجرای `activity` تغییر کند، آن `activity` از بین رفته و مجدداً جهت تطبیق خود با شرایط جدید و بارگذاری منابع جدید، مجدد ایجاد می شود. سیستم اطلاعات `activity` را جهت بازگردانی وضعیت اپلیکیشن به قبل ذخیره کرده و با راه اندازی مجدد `activity`، به این نمونه (`activity`) پاس داده می شود. این اطلاعات که از قبلی ذخیره شده و به `activity` جدید پاس داده می شود، در واقع داخل آبجکت های جاوایی ذخیره می گردد. در ویرایش های قبلی اندروید ذخیره و بازگردانی این آبجکت ها توسط `onRetainNonConfigurationInstance()` و `getLastNonConfigurationInstance()` قابل دسترسی و بازیابی بودند. اما این دو متدهای منسخ هستند و در API های جدید بهتر است برای ذخیره ای آبجکت هایی که می بایست بر اثر تغییر در تنظیمات یا کانفیگ بین نمونه های کلاس `headless fragment` را بدلاً `onRetainNonConfigurationInstance()` استفاده نمایید.

**توجه:** اگر از متدهای `onRetainNonConfigurationInstance()` استفاده می کنید، در آن صورت نباید آبجکتی از `activity` را به نمونه ی بعدی آن (که خود از قبل اشاره گری به آبجکت اولیه دارد) ارسال نمایید. چرا که این کار هدر رفتن حافظه (memory leak) را به دنبال دارد. بدین معنی

که هنوز اشاره گری به آن در garbage collector قادر نخواهد بود نمونه ای از activity را که هنوز اشاره گری به آن در آبجکت جدید وجود دارد، حذف نموده و حافظه را آزاد کند.

## 12-2- مدیریت configuration و تنظیمات

### 12-2- جلوگیری از نابود و مجدد ایجاد شدن activity بر اثر تغییر در تنظیمات و وضعیت برنامه

همان طور که در بالا ذکر شد، activity بر اثر تغییر در تنظیمات و وضعیت دستگاه ( config ) از بین رفته و جهت تطبیق با تنظیمات جدید و همچنین بارگذاری resource های جدید، دوباره ساخته می شود.

تغییر در تنظیمات ممکن است به دنبال یک event که از خود دستگاه اندروید اعلان یا صادر می شود و مربوط به برنامه‌ی جاری می باشد، اتفاق بیافتد.

اطلاعات مربوط به تنظیمات جاری دستگاه در نمونه ای از کلاس Configuration ذخیره می شود. در واقع Configuration یک کلاس است که تمامی اطلاعات مربوط به تنظیمات جاری دستگاه را تعریف می کند و این اطلاعات می توانند منابع و محتوای مورد نیاز اپلیکیشن را تحت تاثیر قرار دهند. کلاس مزبور هم می تواند تنظیمات تعریف شده توسط کاربر (همچون locale list = مقیاس بندی به صورت پویا) را شامل شود و هم تنظیمات دستگاه نظریه حالت دریافت ورودی (input mode)، اندازه‌ی صفحه یا وضعیت و جهت نمایش.

به عنوان مثال زمانی که کاربر با چرخاندن دستگاه، وضعیت نمایش و چیدمان را تغییر می دهد (از نمای عمودی به افقی تغییر می دهد)، در این هنگام سیستم اندروید متوجه می شود که activity جهت تطبیق با شرایط جاری، لازم به بارگذاری منابع جدید دارد. از این‌رو activity کنونی را نابود کرده و بار دیگر با تنظیمات جدید آن راه اندازی می کند.

هنگامی که activity نابود و مجدد ساخته می شود، برنامه نویس می بایست اطمینان حاصل کند که نمونه‌ی جدید activity با اطلاعات مربوط به وضعیت قبلی ایجاد و بارگذاری شود. سیستم

اندروید ابزار و روش های مختلفی برای این منظور ارائه می دهد. در محیط emulator می توانید با فشردن کلیدهای Ctrk+F11 به راحتی تغییر در وضعیت و جهت نمایش را شبیه سازی نمایید.

می توانید activity را طوری در فایل XML تعریف کنید که نسبت به برخی از تغییرات در وضعیت و تنظیمات بی تفاوت باشد و به تبع از راه اندازی مجدد activity در موارد مجبور جلوگیری نمایید. کافی است در فایل تنظیمات AndroidManifest.xml خود، داخل تگ activity، خصیصه configChanges (attribute) را با مقدار مربوطه تنظیم نمایید.

زیر طوری تنظیم شده که در صورت رخداد تغییر در وضعیت نمایش یا موقعیت صفحه کلید (قابل مشاهده/پنهان)، از نو راه اندازی نشود و به بیانی دیگر نسبت به این تغییرات بی تفاوت باشد.

```
<activity android:name=".ProgressTestActivity"
    android:label="@string/app_name"
    android:configChanges="orientation|keyboardHidden|keyboard"
    >
</activity>
```

**توجه:** بهتر است برای مدیریت تغییر در تنظیمات، بجای attribute فوق از روش های دیگر نظری فریم ورک loader یا فاقد UI های fragment (headless fragments) استفاده نمایید. در آینده به شرح مفاهیم نام برده خواهیم پرداخت.

## 12-2- ثابت کردن وضعیت نمایش (orientation) یک activity

این امکان برای شما وجود دارد که activity را در فایل تنظیمات AndroidManifest.xml طوری تعریف کنید که فقط در یک وضعیت یا نمای (orientation) خاص نمایش داده شود. برای مثال فایل XML زیر یک activity تعریف کرده که تنها در حالت و نمای افقی (landscape) به کاربر نشان داده می شود.

```
<activity
    android:name="com.vogella.android.multitouch.MainActivity"
    android:label="@string/app_name"
    android:screenOrientation="landscape" >
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
```

```

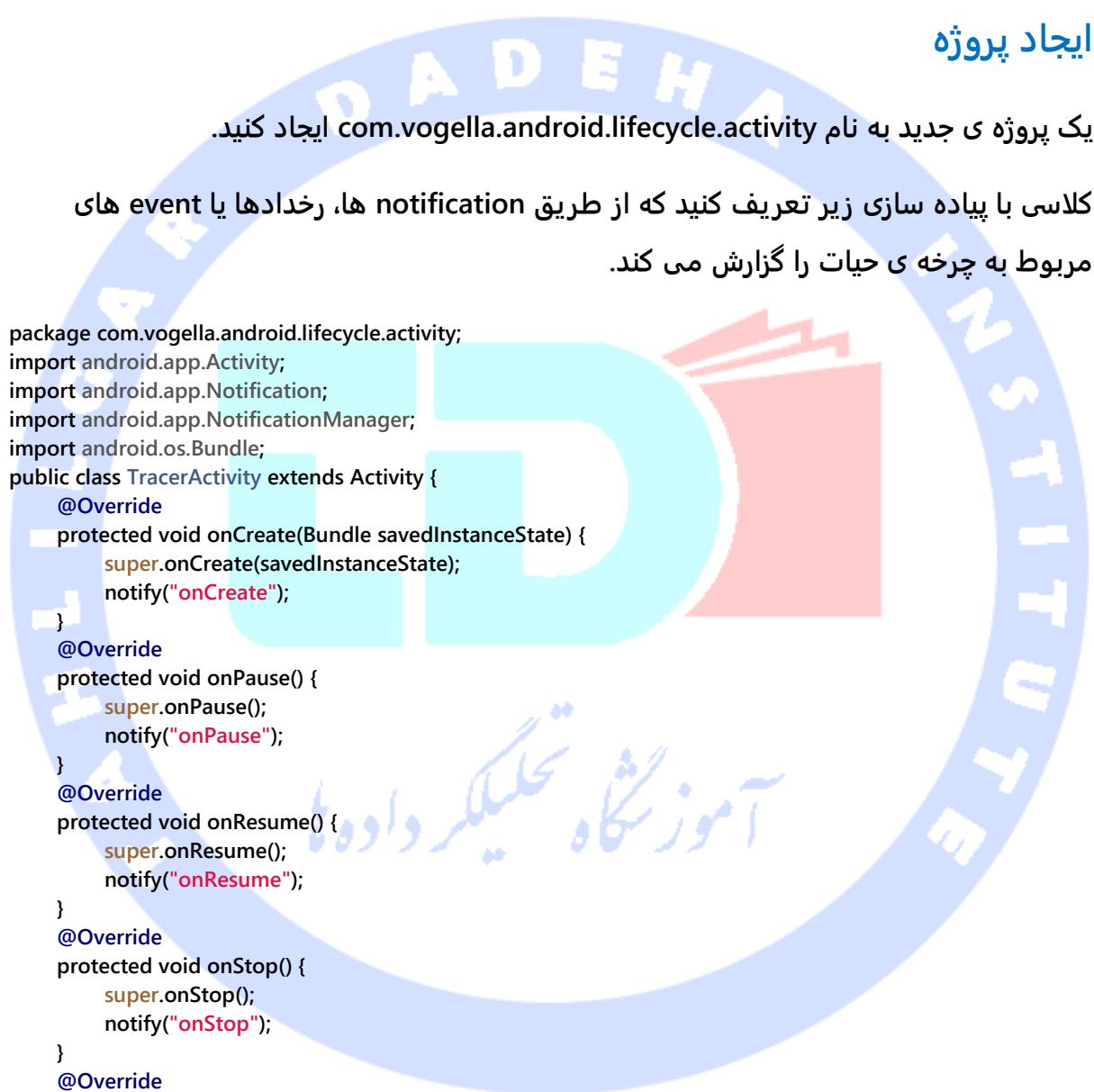
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
```

## Lifecycle: بررسی تمرین

### ایجاد پروژه

یک پروژه‌ی جدید به نام com.vogella.android.lifecycle.activity ایجاد کنید.

کلاسی با پیاده سازی زیر تعریف کنید که از طریق notification ها، رخدادها یا event های مربوط به چرخه‌ی حیات را گزارش می‌کند.



```

package com.vogella.android.lifecycle.activity;
import android.app.Activity;
import android.app.Notification;
import android.app.NotificationManager;
import android.os.Bundle;
public class TracerActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        notify("onCreate");
    }
    @Override
    protected void onPause() {
        super.onPause();
        notify("onPause");
    }
    @Override
    protected void onResume() {
        super.onResume();
        notify("onResume");
    }
    @Override
    protected void onStop() {
        super.onStop();
        notify("onStop");
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
        notify("onDestroy");
    }
    @Override
    protected void onRestoreInstanceState(Bundle savedInstanceState) {
        super.onRestoreInstanceState(savedInstanceState);
    }
}
```

```

        notify("onRestoreInstanceState");
    }
    @Override
    protected void onSaveInstanceState(Bundle outState) {
        super.onSaveInstanceState(outState);
        notify("onSaveInstanceState");
    }
    private void notify(String methodName) {
        String name = this.getClass().getName();
        String[] strings = name.split("\\.");
        Notification noti = new Notification.Builder(this)
            .setContentTitle(methodName + " " + strings[strings.length - 1]).setAutoCancel(true)
            .setSmallIcon(R.drawable.ic_launcher)
            .setContentText(name).build();
        NotificationManager notificationManager =
            (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
        notificationManager.notify((int) System.currentTimeMillis(), noti);
    }
}

```

## ۱۲-۳-۲- ایجاد های مورد نیاز activity

دو دیگر ایجاد کنید که activity بالا را به ارث می بزند. activity اول باید از طریق یک intent، اکتیویتی دوم را راه اندازی کند.

```

package com.vogella.android.lifecycle.activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
public class MainActivity extends TracerActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void onClick(View view) {
        Intent intent = new Intent(this, SecondActivity.class);
        startActivity(intent);
    }
}

```

```

package com.vogella.android.lifecycle.activity;
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
public class SecondActivity extends TracerActivity {
    @Override

```

```

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_second);
}
}

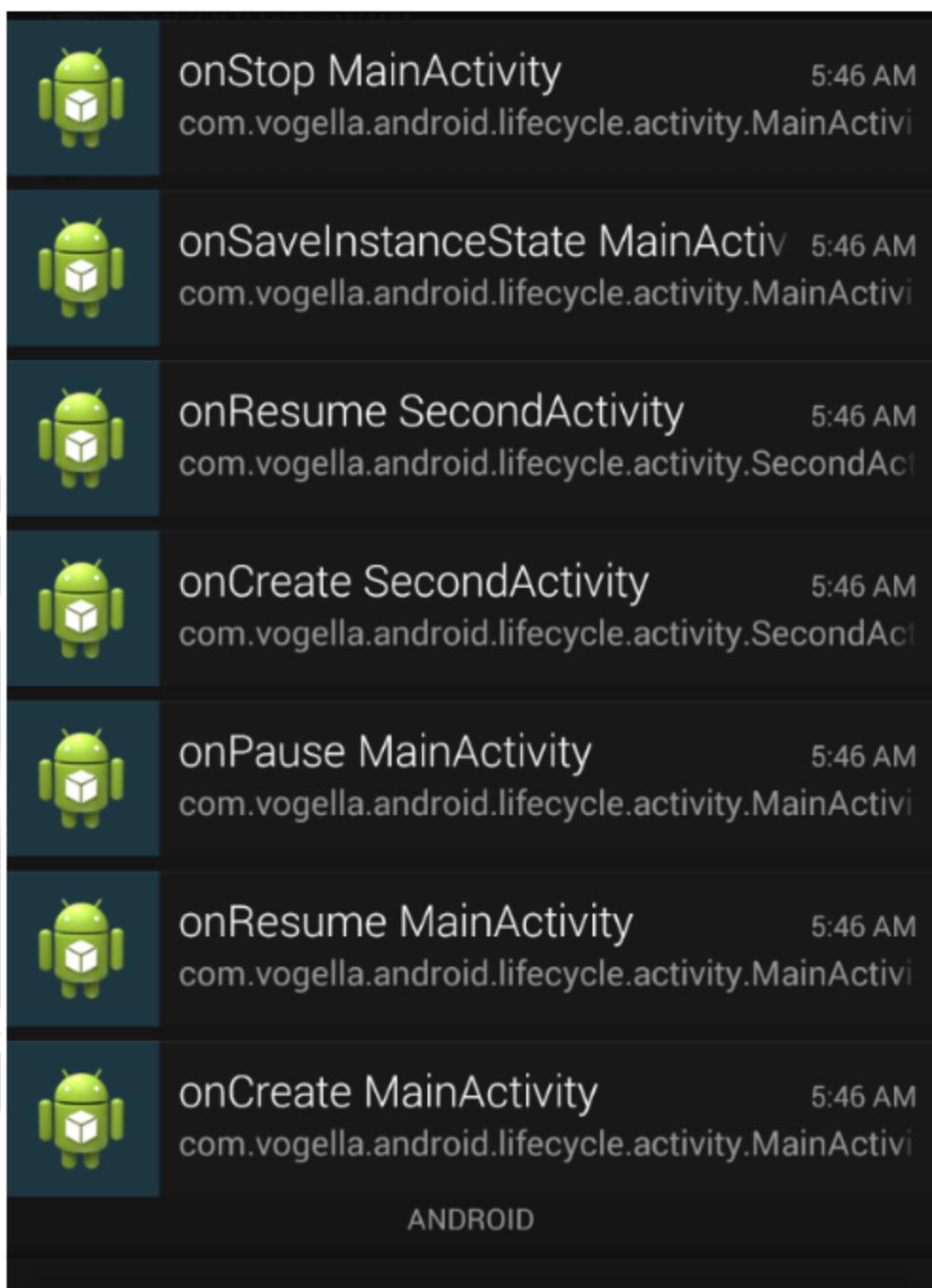
```

**توجه:** بایستی هر دو activity را در فایل تنظیمات (manifest) تعریف نمایید.

## تست اپلیکیشن

اپلیکیشن خود را اجرا کرده و اطمینان حاصل کنید همه چیز طبق برنامه اتفاق می افتد.





پس از راه یابی به activity دوم و مشاهده محتویات آن، دکمه‌ی بازگشت را در این activity (صفحه‌ی دوم اپلیکیشن) فشار دهید. خواهید دید که متدهای `onSaveInstanceState()` و `onCreate()` فراخوانی نشده و به تبع آن وضعیت activity جاری ذخیره نمی‌شود. می‌توانید تشریح کنید چرا این متدهای خورده نمی‌شود!

حال در دوم یا همان صفحه‌ی دوم اپلیکیشن، دکمه‌ی home را فشار دهید. خواهدید دید که به دنبال این رخداد، متدهای onSaveInstanceState() صدا خورده شده و وضعیت activity جاری ذخیره می‌شود. علت فراخوانده شدن این متدها را شرح دهید.

حال یا صفحه‌ی دوم اپلیکیشن را اجرا نمایید. جهت نمایش (orientation) محیط شبیه ساز را تغییر داده و دقت کنید که کدام متدهای مربوط به lifecycle صدا زده می‌شوند. آیا اول نیز دوباره ساخته می‌شود یا فقط activity دوم مجدداً راه اندازی می‌شود.

گزینه‌ی Developer Don't keep activities توسعه دهنده (Options) فعال نمایید. باز دیگر دقت کنید کدام متدها فراخوانده می‌شوند.

## 12-2- بازگردانی وضعیت نمونه activity

ابتدا آرایه‌ای از نوع رشته تعریف کرده و سپس یک آبجکت Spinner (لیست کشویی) به اول خود اضافه نمایید که مقادیر آن از این آرایه پر می‌شود. در کد XML زیر، فایل strings.xml مورد استفاده‌ی اول را مشاهده می‌کنید.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Lifecycle</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>
    <string-array name="operating_systems">
        <item>Ubuntu</item>
        <item>Android</item>
        <item>iOS</item>
    </string-array>
</resources>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >
    <Spinner
```

```

    android:id="@+id/spinner"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginRight="58dp"
    android:entries="@array/operating_systems" />
<Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="onClick"
    android:layout_gravity="bottom"
    android:text="Start new Activity" />
</LinearLayout>

```

اندروید وضعیت و اطلاعات مربوط به آبجکت Spinner یا همان لیست کشویی را به صورت خودکار بازگردانی می کند. در واقع سیستم اندرود خود وضعیت جاری آبجکت مزبور (همچون انتخاب کاربر) را ذخیره کرده و در صورت تغییر در تنظیمات اپلیکیشن یا دستگاه و همچنین راه اندازی مجدد activity ها (نابود شدن activity و بازسازی آن)، آن را به وضعیت قبلی بازگردانی می کند.

حال مقادیر ثابت آرایه در فایل layout را حذف کرده و آن را در کد برنامه (از طریق source code) به آبجکت Spinner تخصیص دهید.

```

// configure the spinner in code
Spinner spinner = (Spinner) findViewById(R.id.spinner);
String[] values = getResources().getStringArray(R.array.operating_systems);
ArrayAdapter<String> adapter =
    new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, values);
spinner.setAdapter(adapter);

```

مقادیر تخصیص یافته ی ثابت را از فایل layout خود نیز حذف نمایید.

```

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >
<Spinner
    android:id="@+id/spinner"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"

```

```
        android:layout_marginRight="58dp"
    />
<Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="onClick"
    android:layout_gravity="bottom"
    android:text="Start new Activity" />
</LinearLayout>
```

اطمینان حاصل نمایید که وضعیت آبجکت Spinner همچنان مانند قبل به صورت خودکار بازگردانی می شود.



### بخش سوم :

## 13-2-مبحث امنیت و مجوزهای دسترسی در اندروید (Security&Permission)

### 13-2-1-مفهوم امنیت در اندروید

سیستم اندروید به هنگام نصب یک اپلیکیشن بر روی دستگاه، یک **GID** (شناسه ی یکتا که به گروه کاربری خاص تخصیص می یابد) و **UID** (شناسه ی یگانه ی کاربری) به آن اپلیکیشن اختصاص می دهد. در واقع تمامی فایل های یک آپلیکیشن در انحصار این کاربر یا به اصطلاح **private** هستند و به تبع دیگر اپلیکیشن ها اجازه ی دسترسی به این فایل ها را ندارند. علاوه بر آن، هر اپلیکیشن اندروید تحت فرایند اختصاصی خود اجرا و راه اندازی می شود. از اینرو، به واسطه ی این ویژگی هسته ی Linux، هر اپلیکیشن اندرویدی از دیگر برنامه های کاربردی در حال اجرا کاملا مستقل و مجزا است. در صورتی هم که یک اپلیکیشن اندروید می بایست بر حسب نیاز اطلاعاتی را با یک اپلیکیشن دیگر رد و بدل کرده و آن ها را به اشتراک بگذارد، این اتفاق بایستی به صورت صریح و توسط یک کامپوننت نرم افزاری که وظیفه ی آن انتقال داده می باشد همچون یک **service** یا **content provider** صورت گیرد.

### 13-2-2-مفهوم مجوزهای دسترسی در اندروید

اندروید از یک سیستم تخصیص مجوز برخوردار است که ویژه ی هر **task** یا عملیات مجوزها و سطوح دسترسی خاصی معین می کند. در حقیقت هر اپلیکیشن می تواند جهت انجام عملیات خاص، مجوزهای لازم را از این سامانه درخواست نماید. برای مثال، یک برنامه ی کاربری می تواند از سیستم تخصیص مجوز، درخواست دسترسی به اینترنت را داشته باشد.

سیستم مزبور سطوح مختلفی برای مجوزها دارد که به آن در اصطلاح **protection level** گفته می شود.

لازم به توضیح است که مفهوم مجوز از کتابخانه های اندروید، نسخه ی 23 (API level) تغییر کرده است. قبل از ویرایش مذکور، یک اپلیکیشن حین نصب مجوزهای مورد نیاز و درخواست خود را به

کاربر اعلان می کرد. اما از API 23 به بعد دیگر تنها در زمان اجرای برنامه مجوزهای لازم را از کاربر درخواست می کند.

اپلیکیشن های اندروید مجوزهای مورد نیاز خود را در فایل تنظیمات اندروید (manifest) اعلان می کنند. البته اپلیکیشن می تواند مجوزهای اضافی در این فایل تعریف کند که دسترسی به برخی کامپوننت های نرم افزار را محدود می سازند.

### 13-2-سیستم مدیریت و تخصیص مجوز در اندروید پیش از API 23

قبل از ویرایش 23 کتابخانه های اندروید، مجوزهای مورد نیاز قبل از نصب اپلیکیشن از کاربر درخواست می شدند. کاربر تصمیم می گرفت که آیا مجوزهای مربوطه باید به برنامه داده شوند یا خیر. چنانچه کاربر مجوز مورد درخواست اپلیکیشن را تایید نمی کرد، در آن صورت اپلیکیشن مورد نظر به هیچ وجه بر روی دستگاه نصب نمی شد. اگر مجوز لازم را حین نصب به اپلیکیشن می داد، اپلیکیشن به بخش های مد نظر خود دسترسی پیدا می کرد و دیگر کاربر نمی توانست این امر را (پس از اتمام نصب برنامه) تغییر دهد.

### 13-2-سیستم تخصیص مجوز از API 23 به بعد (اعطای مجوز به هنگام اجرا برنامه)

ویرایش 6.0 سیستم عامل اندروید، Marshmallow، یک permission model جدید را معرفی کرد که طی آن مجوزهای لازم، در زمان اجرای اپلیکیشن از کاربر درخواست می شود. بنابراین اگر اپلیکیشنی که در دست توسعه دارد، ویژه‌ی این نسخه طراحی شده (targetsdkversion) آن را بر روی 23 تنظیم کرده اید، لازم است از مدل تخصیص مجوز جدید استفاده نمایید.

در اندروید دو سطح دسترسی و امنیت به شرح زیر تعریف شده است:

- Normal: در این سطح، مجوزهایی درخواست می شود که به اطلاعات، فایل ها شخصی کاربر یا فعالیت و عملیات دیگر اپلیکیشن ها لطمه ای وارد نمی کند (به بخش های حساس

دسترسی ندارد). به عنوان مثال می توان به مجوز تنظیم منطقه ی زمانی اشاره کرد. مجوزهایی که در این سطح هستند معمولاً به صورت خودکار به اپلیکیشن اعطا می شوند.

- **Dangerous**: مجوزهای مربوط به این سطح، اغلب با اطلاعات شخصی کاربر سروکار دارند و حتی ممکن است این اطلاعات یا فرایندهای مربوط به اپلیکیشن های دیگر را دستکاری کنند. به عنوان مثال می توان به اجازه ی دسترسی به اطلاعات مخاطبین کاربر اشاره کرد. این دست از مجوزها باید در زمان اجرای اپلیکیشن از کاربر درخواست می شوند.

برای بررسی اینکه آیا مجوز خاصی به اپلیکیشن داده شده یا خیر، می توان از متد `checkSelfPermission`

```
// called in a standard activity, use ContextCompat.checkSelfPermission for AppCompatActivity
int permissionCheck = checkSelfPermission(this, Manifest.permission.WRITE_CALENDAR);
if (!permissionCheck == PackageManager.PERMISSION_GRANTED) {
    // User may have declined earlier, ask Android if we should show him a reason
    if (shouldShowRequestPermissionRationale(thisActivity, Manifest.permission.WRITE_CALENDAR)) {
        // show an explanation to the user
        // Good practise: don't block thread after the user sees the explanation, try again to request the
        permission.
    } else {
        // request the permission.
        // CALLBACK_NUMBER is a integer constants
        requestPermissions(thisActivity, new String[]{Manifest.permission.WRITE_CALENDAR},
        CALLBACK_NUMBER);
        // The callback method gets the result of the request.
    }
} else {
    // got permission use it
}
```

هنگامی که از کاربر درخواست اعطای مجوز می شود، شما یک `callback` دریافت می کنید.

#### @Override

```
public void onRequestPermissionsResult(int requestCode, String permissions[], int[] grantResults) {
    switch (requestCode) {
        case MY_PERMISSIONS_REQUEST_READ_CONTACTS: {
            // If request is cancelled, the result arrays are empty.
            if (grantResults.length > 0
                && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                // permission was granted, do your work....
            } else {
                // permission denied
                // Disable the functionality that depends on this permission.
            }
        }
    }
}
```

```
        return;  
    }  
    // other 'case' statements for other permissions  
}  
}
```



## 14-پیاده سازی list یا grid جهت نمایش آیتم ها در اندروید

پیاده سازی list (نمای فهرستی) و grid (نمای جدولی و خانه بندی شده همراه با سطر و ستون) در اپلیکیشن های اندروید به وسیله هی کلاس / ویجت RecyclerView

این آموزش نحوه استفاده از RecyclerView کامپوننت رابط کاربری (جهت پیاده سازی لیست های پیچیده) در اپلیکیشن های مبتنی بر اندروید را به شما آموزش می دهد.

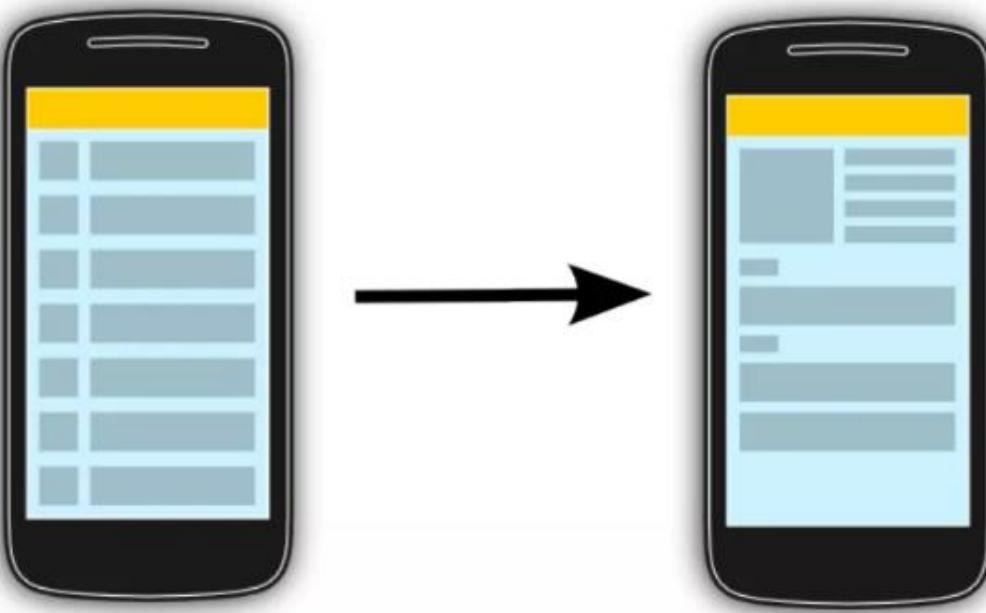
### 14-2-استفاده از grid یا list در اپلیکیشن اندروید

نمایش آیتم ها در قالب یک list (با نمای فهرستی) یا grid (با نمای جدولی و به صورت خانه بندی شده) در اپلیکیشن های تحت موبایل یک الگوی کاملاً معمول و پرکاربرد است که به فور شاهد آن هستید. کاربر در مقابل خود مجموعه ای از آیتم های مرتب و سازماندهی شده را می بیند که به راحتی به وسیله هی یک نوار اسکرول قابل پیمایش می باشد. حال این مجموعه می تواند یک list، grid یا هر ساختار سازمان دهی شده هی دیگری از داده ها باشد. تصویر زیر یک activity با این نما را به نمایش می گذارد.



کاربر با این مجموعه از آیتم ها از طریق event هایی که با لمس نمایشگر اتفاق می افتد (touch event) یا نوارابزار (toolbar) تعامل می کند. هر یک از این المان ها می تواند برای کاربر قابل انتخاب باشد. در پی کلیک کاربر بر روی یک المان، ممکن است toolbar بروز رسانی شود یا به طور کلی یک صفحه ی جدید با جزئیات فراوان به نمایش در آید.

## آموزشگاه تحلیلکرداوه



**۱۴-۲-استفاده از RecyclerView** جهت پیاده سازی لیست های پیچیده  
یک کلاس است که به برنامه نویس قابلیت پیاده سازی لیست های پیچیده را می دهد. این لیست می تواند حجم بالایی از اطلاعات را نمایش داده که ممکن است در زمان اجرای برنامه (بر اساس افعال کاربر) بارها تغییر کنند.

RecyclerView در واقع نسخه‌ی جدیدی از GridView و ListView است که توسط چارچوب نرم افزاری (framework) اندروید در اختیار توسعه دهنده قرار می‌گیرد. RecyclerView تعدادی زیادی از مشکلات و کاستی‌هایی که در widget‌های قبلی وجود داشت را برطرف نموده و سبکی از برنامه نویسی را اعمال می‌کند که افزایش کارایی و سرعت در اجرای اپلیکیشن را به دنبال دارد. علاوه بر موارد ذکر شده، این کلاس همراه با اینیمیشن‌های پیش فرض و از قبل تعریف شده ارائه می‌گردد که المان‌ها را با اینیمیشن از لیست حذف کرده یا به آن اضافه می‌کند.

به شما اجازه می‌دهد تا از RecyclerView های مختلف برای مدیریت چیدمان و موقعیت دهی آیتم‌ها استفاده نمایید. RecyclerView برای پیاده سازی لیست و نمایش آیتم‌ها با سه کلاس همکاری دارد. ViewHolder که وظیفه‌ی نگهداری یک المان در لیست را برعهده دارد

(تمامی المان هایی که در لیست مشاهده می کنید، اشاره گر یا متغیری متناظر در آبجکت view دارند). دومین المان، کلاس adapter هست که همان المانی که در لیست قرار می گیرد را ساخته، آن را در viewholder قرار می دهد و در نهایت داده ها را به هر view متصل یا bind می کند. و اما آخرین المان LayoutManager هست که چگونگی چیدمان و نمایش المان ها در لیست را مدیریت می کند (لیست را با نمای افقی، عمودی، به صورت جدول بندی شده نمایش دهد).

یک کلاس static و درون ساخته در adapter است که اشاره گری (reference) و ViewHolder متغیری متناظر (view) به view مربوطه نزد خود نگه می دارد. با نگه داشتن اشاره گر به این view ها در کلاس مزبور، توسعه دهنده می تواند از فراخوانی مکرر متد findViewById() جهت بروز رسانی widget ها با داده های جدید خودداری کند (که در نهایت صرفه جویی در منابع و افزایش کارایی اپلیکیشن را در پی دارد).

## Adapter-2-14-3

همان طور که در بالا اشاره شد، دومین المان، کلاس adaptor هست که view یا همان المانی که در لیست قرار می گیرد را ساخته، آن را در آبجکت viewholder قرار می دهد و در نهایت داده ها را به هر view متصل یا bind می کند. کلاس adapter در واقع وظیفه‌ی مدیریت data model را بر عهده داشته و آن را با توجه به هر المان در لیست (ویجت RecyclerView) تنظیم یا بروز رسانی می کند. کلاس نام برده ویژگی ها و متد های کلاس RecyclerView.Adapter را به ارث برده (این کلاس را extend می کند) و توسط متد recycler view به RecyclerView.setAdapter تخصیص adapter می یابد. ورودی adapter می تواند هر آبجکت جاوایی باشد. بر پایه‌ی این ورودی، آبجکت recycler view از طریق متد getCount() بازگردانی نماید.

می دانید adapter چگونه چیدمان و ظاهر (layout) آیتم های موجود در لیست را مدیریت و آماده می کند؟ در جواب باید گفت که adapter برای نمایش هر آیتم در لیست، فایل layout متناظر و مربوطه‌ی آن را inflate می کند. (= عبارت است از تبدیل فایل LAYOUT XML به آبجکت view جهت قرار گرفتن در کلاس ACTIVITY). در واقع هر زمان که بخواهید layout را برای مثال در زمان اجرا در برنامه نمایش دهید، باید آن را inflate نمایید). این کار در متدهای onCreateViewHolder() و onBindViewHolder() انجام می شود. این متدهای از اینکه هر المان بصری داخل recycler view آبجکتی از نوع ViewHolder را به عنوان خروجی بازمی گردانند.

به واسطه‌ی این نمونه (instance) می توانید به view ها در layout ها بازگذاری و به صورت نهایی ارائه / inflate شده دسترسی داشته باشید. هر زمان که لازم باشد یک view جدید ساخته شود، متدهای onCreateViewHolder() و onBindViewHolder() این view را می سازد، سپس آن را در ViewHolder قرار می دهد و در نهایت آبجکت مورد نظر را برای نمایش به RecyclerView می دهد.

هر المان در لیست (recycler view) را با آیتم متناظر و مناسب (dr) پر Adapter می کند. پس از اینکه داده‌ی (data item) مورد نظر در دسترس قرار می گیرد، adapter این داده را به تک تک widget های مربوطه که قبلاً inflate و در حافظه بازگذاری کرده، متصل یا bind می کند. این عملیات در بدنه‌ی متدهای onBindViewHolder() و onBindViewHolder() انجام می شود. به عبارت دیگر این متدهای view را می گیرد و داده‌های مربوطه از data model را به آن به صورت دو طرفه وصل می کند.

برای مثال، آیتمی را در لیست تصور کنید که ممکن است یک عکس در سمت چپ و دو خط متن مانند تصویر زیر در وسط داشته باشد.



فایل layout متناظر این سطر به صورت زیر خواهد بود.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="?android:attr/listPreferredItemHeight"
    android:padding="6dip" >
    <ImageView
        android:id="@+id/icon"
        android:layout_width="wrap_content"
        android:layout_height="fill_parent"
        android:layout_alignParentBottom="true"
        android:layout_alignParentTop="true"
        android:layout_marginRight="6dip"
        android:contentDescription="TODO"
        android:src="@drawable/ic_launcher" />
    <TextView
        android:id="@+id/secondLine"
        android:layout_width="fill_parent"
        android:layout_height="26dip"
        android:layout_alignParentBottom="true"
        android:layout_alignParentRight="true"
        android:layout_toRightOf="@+id/icon"
        android:ellipsize="marquee"
        android:singleLine="true"
        android:text="Description"
        android:textSize="12sp" />
    <TextView
        android:id="@+id/firstLine"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_above="@+id/secondLine"
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true"
        android:layout_alignWithParentIfMissing="true"
        android:layout_toRightOf="@+id/icon"
        android:gravity="center_vertical"
        android:text="Example application"
        android:textSize="16sp" />
</RelativeLayout>
```

## اضافه کردن dependency (کتابخانه ها) به فایل gradle build جهت استفاده از recycler view

ویجت RecyclerView توسط یک کتابخانه ی مجزا در اختیار توسعه دهنده قرار گرفته و در واقع جدا از فریم ورک اندروید ارائه می شود که از ورژن 7 کتابخانه های اندروید به بعد ( API 7 ) و اعلی های

بالاتر) قابل استفاده می باشد. به منظور استفاده از recycler view در پروژه‌ی خود، لازم است مربوطه را مانند زیر به فایل gradle build اضافه نمایید.

```
dependencies {
```

```
    ...
    compile "com.android.support:recyclerview-v7:23.0.1"
}
```

## پیش فرض Layout manager-2-14-4

تعیین می کند که داده ها چگونه در لیست یا همان RecyclerView به نمایش در آیند (نحوه‌ی چیدمان و آرایش المان‌ها در لیست را مدیریت می کند). کتابخانه‌ی recycler یک سری layout manager درون ساخته و پیش فرض ارائه می دهد که در زیر عنوان شده:

- آیتم‌ها را در یک لیست عمودی یا افقی قابل پیمایش (دارای نوار اسکرول) به نمایش می گذارد.
- آیتم‌ها را با نمای جدولی (به صورت خانه بندی شده) و به اصطلاح در یک grid نمایش می دهد.
- آیتم‌ها را در قالب یک staggered grid به نمایش می گذارد (در grid View معمولی، هر گرید یا خانه برای خود طول و ارتفاع متفاوتی می تواند داشته باشد و به عبارتی خانه‌های جدول و آیتم‌هایی که در لیست به نمایش در می آیند می توانند نامتقارن باشند).

## Recycler View-2-14-5

به منظور پیاده سازی RecyclerView لازم است تعدادی کلاس را قبلش پیاده سازی کرده باشید. در جدول زیر به برخی از مهمترین این کلاس‌ها همراه با شرح کاربرد اشاره شده است.

کلاس	کاربرد	Optional
Adapter	view یا همان المانی که در لیست قرار می گیرد را ساخته، آن را در قرار می دهد و در نهایت داده ها را به هر view متصل یا bind می کند.	الزامي
ViewHolder	تمامی المان هایی که در لیست مشاهده می کنید، اشاره گر یا متغیری متناظر در آبجکت view holder دارند. این متغیرها با مقدار المان های لیست پر می شوند. به عبارت بهتر این کلاس تک تک view های تشکیل دهنده ی لیست که متناظر آن ها به صورت تگ در فایل قرار دارد را در خود ذخیره کرده و به طور آماده در دسترس قرار می دهد، به گونه ای که دیگر لازم نباشد برای هر با استفاده از layout مورد نظر، در فایل آن view را جستجو کنید.	الزامي
LayoutManager	این کلاس مدیریت نحوه ی چیدمان و نمایش المان ها در لیست را بر عهده دارد. به عبارت دیگر view آیتم ها یا LayoutManager های موجود در لیست را اندازه گیری و موقعیت دهی کرده و همچنین قوانین مربوط به زمان بازیافت و استفاده ی مجدد از view هایی که دیگر برای کاربر قابل مشاهده نیستند را تعیین می کند. با ویرایش LayoutManager، کلاس RecyclerView قادر خواهد بود به راحتی یک لیست افقی قابل بیمایش، یک grid معمولی، staggered grid، لیست های افقی با قابلیت بیمایش و غیره ... بیاده سازی کند.	الزامي است، هرچند پیاده ساری های بیش فرض و آماده دارد.

کلاس	کاربرد	Optional
ItemDecoration	بر روی آیتم های لیست یا پیرامون آن ها decoration (خط های تزئین/خط پیرامون یا جداکننده غیره ..) ترسیم می کند.	دارای رفتار پیش فرض است، هرچند پیاده سازی یا بدنه ی آن قابل بازنویسی یا override می باشد.
ItemAnimator	این امکان را فراهم می کند تا برای مبنای عملیاتی که بر روی آیتم های لیست اجرا می شود (یک المان اضافه/حذف شده یا ترتیب آن تغییر می کند)، اینیمیشن خاصی تعریف نمایید. به عبارت دیگر به شما اجازه می دهد تا اینیمیشن هایی تعریف کنید که با تغییر در adapter، بر روی المان های لیست اعمال می شوند.	دارای رفتار پیش فرض بوده، اما قابل بازنویسی می باشد.

می توانید این animation ها و layout manager ها را به صورت اختصاصی پیاده سازی نمایید.

## ۱۴-۲- مدیریت event های مربوط به کلیک یا لمس نمایشگر در recycler view

رخدادهایی که بر اثر لمس یا کلیک بر روی آیتم های موجود در لیست فعال می شوند، بایستی توسط view ها در recycler view مدیریت شود. در صورتی که view سبب فراخوانی رخدادی در object میزبان (activity یا fragment) شود، می توانید آن را از طریق متدهای سازنده (constructor) کلاس adapter مدیریت نمایید. این کار به شما امکان می دهد تا اشاره گری (reference) به آبجکت مورد نظر ذخیره کرده و متدهای آن را برای بازخورد صدا بزنید.

## ۱۴-۷-۲- استفاده از layout های مختلف در recycler view

می بایست به ازای هر سطر در لیست، یک layout ایجاد کند. root فایل layout معمولاً یک کلاس ViewGroup است که خود چندین view مانند TextView و ImageView را شامل می شود. تصویر زیر لیستی را به نمایش می گذارد که در آن سطرهای زوج و فرد چیدمان و layout متفاوتی دارند.



نوع داده ای view را از طریق متد getItemViewType تشخص می دهد (متد getItemViewType مستقر در مکان خاص را به منظور recycle و استفاده ای مجدد از آن برمی گرداند). چارچوب نرم افزاری اندروید (framework) خود به صورت پیش فرض متد onCreateViewHolder را، در صورتی که برای این نوع داده ای لازم باشد، صدا می زند. در متد مذکور، layout مناسب را برای نوع مورد نظر inflate کرده و سپس view holder مربوطه را بازگردانی می کنید.

## 14-2-پیاده سازی انیمیشن های اختصاصی

به منظور تنظیم سفارشی و پیاده سازی انیمیشن ها در کلاس RecyclerView، می بایست از کلاس RecyclerView ارث بری کرده (آن را extend نمایید) و سپس به وسیله متد RecyclerView.setItemAnimator() آن را به widget یا آیتم مورد نظر در لیست خود تخصیص دهید.

## 14-2-فیلتر و مرتب سازی داده ها

کلاس adapter عملیات فیلتر و مرتب سازی داده ها را بر عهده دارد. برای این منظور لازم است منطق مربوطه را در بدنه ای کلاس adapter اختصاصی خود پیاده سازی نمایید.

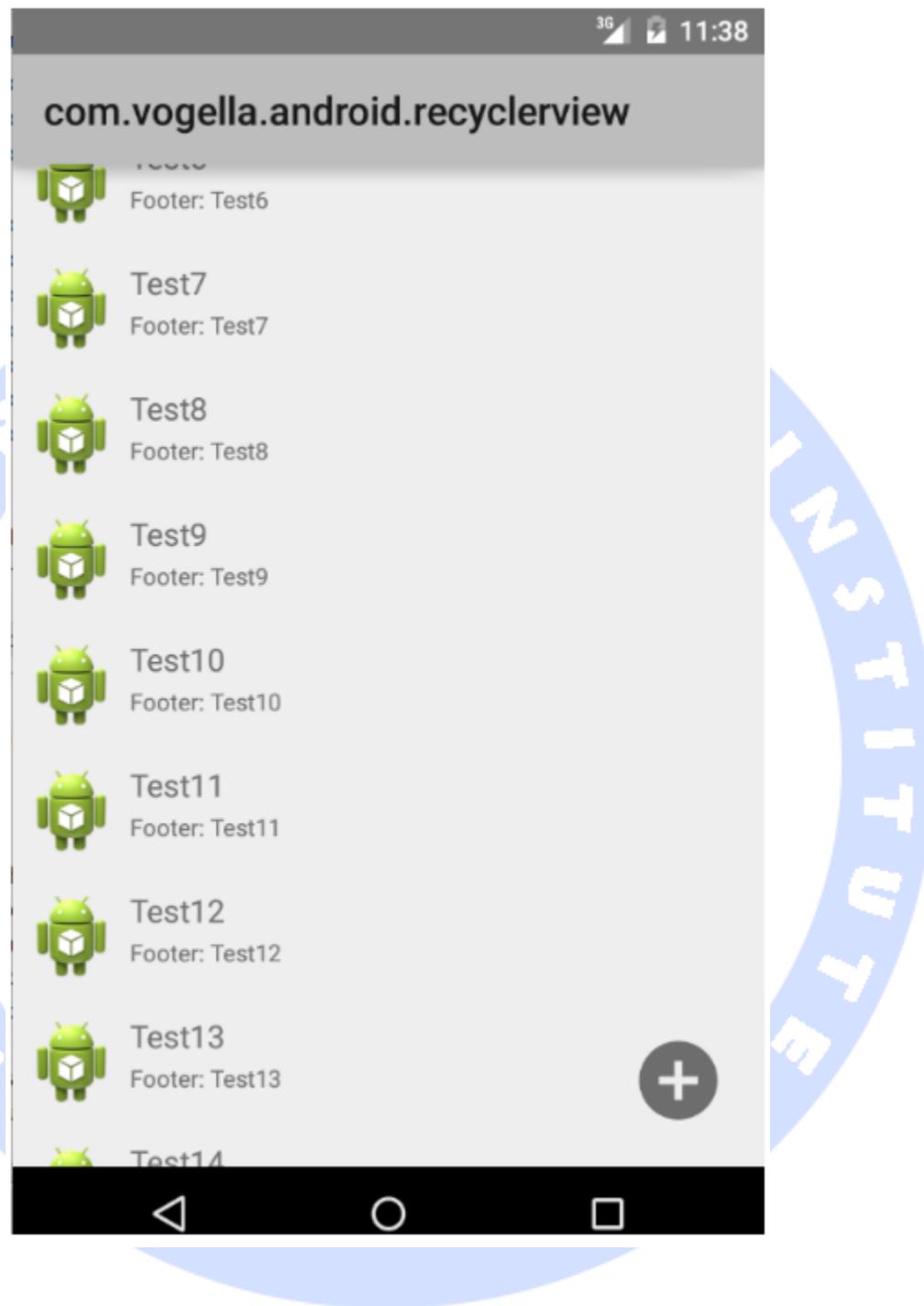
## 14-2-بروز رسانی داده ها در کلاس adapter

به وسیله ای متد notifyItemInserted(position) در کلاس adapter می توان view را مطلع نمود که یک المان جدید در مکان قرار گیری خاصی درج شده است.

به وسیله ای متد notifyItemRemoved(position); در مکان خاص، از view حذف شده است.

## تمرین: پیاده سازی RecyclerView و استفاده از آن در پروژه

در این تمرین یک پروژه ایجاد خواهید نمود که با استفاده از کلاس RecyclerView لیست پیچیده ای ایجاد نموده و برای کاربر نمایش می دهید.



## 14-2- ایجاد پروژه ها و افزودن dependency مورد نظر در فایل

### gradle build

ابتدا یک پروژه جدید Android با نام پکیج com.vogella.android.recyclerview ایجاد نمایید.

جهت استفاده از کلاس `RecyclerView`, کتابخانه یا `dependencies` زیر را در بخش `Dependency` کتابخانه یا `dependencies` در `gradle build` اضافه نمایید.

```
dependencies {
    ...
    compile "com.android.support:recyclerview-v7:23.0.1"
}
```

## 14-2-ایجاد فایل های layout مورد نیاز

یک فایل `layout` به نام `activity_main.xml` ایجاد کرده و سپس `RecyclerView` را در آن تعریف نمایید.

```
<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="${relativePackage}.${activityClass}" >
    
    <android.support.v7.widget.RecyclerView
        android:id="@+id/my_recycler_view"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scrollbars="vertical" />
    <ImageView
        android:id="@+id/imageView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentRight="true"
        android:layout_marginBottom="12dp"
        android:layout_marginRight="12dp"
        android:elevation="2dp"
        android:src="@drawable/ic_add_circle" />
</RelativeLayout>
```

**نکته** : همان طور که در فایل XML می بینید، المان `ImageView` یک `attribute` به نام `android:elevation` دارد که با پارامتر `2dp` مقدار دهنده است. این ویژگی به اندروید دستور می دهد که برای المان مذبور سایه ایجاد کند.

حال یک فایل `layout` تعریف می کنید که ویژگی های هر آیتم در لیست را مشخص می کند.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="?android:attr/listPreferredItemHeight"
```

```

    android:padding="6dip" >
<ImageView
    android:id="@+id/icon"
    android:layout_width="wrap_content"
    android:layout_height="fill_parent"
    android:layout_alignParentBottom="true"
    android:layout_alignParentTop="true"
    android:layout_marginRight="6dip"
    android:contentDescription="TODO"
    android:src="@drawable/ic_launcher" />
<TextView
    android:id="@+id/secondLine"
    android:layout_width="fill_parent"
    android:layout_height="26dip"
    android:layout_alignParentBottom="true"
    android:layout_alignParentRight="true"
    android:layout_toRightOf="@+id/icon"
    android:ellipsize="marquee"
    android:singleLine="true"
    android:text="Description"
    android:textSize="12sp" />
<TextView
    android:id="@+id/firstLine"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_above="@+id/secondLine"
    android:layout_alignParentRight="true"
    android:layout_alignParentTop="true"
    android:layout_alignWithParentIfMissing="true"
    android:layout_toRightOf="@+id/icon"
    android:gravity="center_vertical"
    android:text="Example application"
    android:textSize="16sp" />
</RelativeLayout>

```

**نکته:** در حال حاضر نمی توان لیستی که با کلاس RecyclerView پیاده سازی شده را در حالت پیش نمایش (preview) مشاهده نمود.

کلاس adapter را به صورت زیر پیاده سازی نمایید.

```

package com.vogella.android.recyclerview;
import java.util.ArrayList;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup;
import android.widget.TextView;
public class MyAdapter extends RecyclerView.Adapter<MyAdapter.ViewHolder> {

```

```

private ArrayList<String> mDataset;
// Provide a reference to the views for each data item
// Complex data items may need more than one view per item, and
// you provide access to all the views for a data item in a view holder
public class ViewHolder extends RecyclerView.ViewHolder {
    // each data item is just a string in this case
    public TextView txtHeader;
    public TextView txtFooter;
    public ViewHolder(View v) {
        super(v);
        txtHeader = (TextView) v.findViewById(R.id.firstLine);
        txtFooter = (TextView) v.findViewById(R.id.secondLine);
    }
}
public void add(int position, String item) {
    mDataset.add(position, item);
    notifyItemInserted(position);
}
public void remove(String item) {
    int position = mDataset.indexOf(item);
    mDataset.remove(position);
    notifyItemRemoved(position);
}
// Provide a suitable constructor (depends on the kind of dataset)
public MyAdapter(ArrayList<String> myDataset) {
    mDataset = myDataset;
}
// Create new views (invoked by the layout manager)
@Override
public MyAdapter.ViewHolder onCreateViewHolder(ViewGroup parent,
    int viewType) {
    // create a new view
    View v = LayoutInflater.from(parent.getContext()).inflate(R.layout.rowlayout, parent, false);
    // set the view's size, margins, paddings and layout parameters
    ViewHolder vh = new ViewHolder(v);
    return vh;
}
// Replace the contents of a view (invoked by the layout manager)
@Override
public void onBindViewHolder(ViewHolder holder, int position) {
    // - get element from your dataset at this position
    // - replace the contents of the view with that element
    final String name = mDataset.get(position);
    holder.txtHeader.setText(mDataset.get(position));
    holder.txtHeader.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            remove(name);
        }
    });
    holder.txtFooter.setText("Footer: " + mDataset.get(position));
}

```

```

    }
    // Return the size of your dataset (invoked by the layout manager)
    @Override
    public int getItemCount() {
        return mDataset.size();
    }
}

```

اکنون شما می توانید activity را در recycler view خود تنظیم نمایید.

```

public class MyActivity extends Activity {
    private RecyclerView mRecyclerView;
    private RecyclerView.Adapter mAdapter;
    private RecyclerView.LayoutManager mLayoutManager;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.my_activity);
        mRecyclerView = (RecyclerView) findViewById(R.id.my_recycler_view);
        // use this setting to improve performance if you know that changes
        // in content do not change the layout size of the RecyclerView
        mRecyclerView.setHasFixedSize(true);
        // use a linear layout manager
        mLayoutManager = new LinearLayoutManager(this);
        mRecyclerView.setLayoutManager(mLayoutManager);
        // specify an adapter (see also next example)
        mAdapter = new MyAdapter(myDataset);
        mRecyclerView.setAdapter(mAdapter);
    }
    ...
}

```

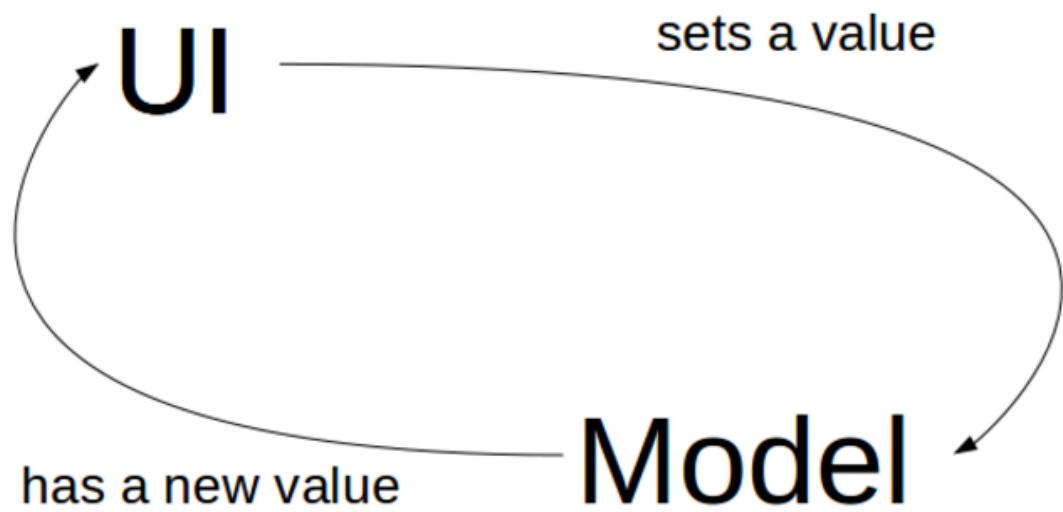
بخش پنجم :

## استفاده از کتابخانه `data binding` جهت اتصال دو طرفه داده ها از `Model /UI` به

بر کسی پوشیده نیست که اپلیکیشن های اندروید دائما با داده سروکار دارند. این داده ها قاعدها باید یا از یک منبع داده همچون دیتابیس خوانده شوند یا از اینترنت گرفته شده و در اپلیکیشن مورد نظر برای کاربر به نمایش در آیند.

طبعتا داده های برنامه باید با نما و ظاهر خاصی در نمایشگر دستگاه نشان داده شوند. همان طور که می دانید در اندروید توسعه دهنده اغلب با آبجکت های activity، fragment و لایه ها سروکار دارد و ظاهر برنامه را با آیتم های ذکر شده طراحی می کند. در مرحله ی بعدی توسعه دهنده این با جاوا این آبجکت ها را مقداردهی می کند. نا گفته مشخص است که این کار بسیار تکراری است و در فایل های XML انجام می شود. به این دلیل Google کتابخانه ای ارائه داد که به واسطه ی آن برنامه نویس قادر است به راحتی مقداردهی را در فایل های layout و با همان تگ های XML انجام دهد و سپس مقادیر به view های مربوطه متصل شوند. این کتابخانه یک support library هست که با وجود جدید بودن، به راحتی برای ویرایش های قبلی، البته از 2.1 به بالا قابل استفاده می شود.

این مبحث به شما آموزش می دهد چگونه با استفاده از کتابخانه ی Data Binding فایل های layout با فرمت XML بسازید و کد لازم برای متصل کردن منطق اپلیکیشن و فایل های layout را به حداقل برسانید.



جهت استفاده از این کتابخانه، لازم است افزونه `androidx.databinding` ویرایش 1.5.0 سیستم کامپایل Gradle یا بالاتر را نصب نمایید.

به منظور استفاده از این کتابخانه برای اتصال داده، می بایست تکه کد زیر را به فایل `app/build.gradle` اضافه نمایید:

```

android {
    ...
    dataBinding {
        enabled = true
    }
}
  
```

## 15-2- فایل های layout اتصال داده از UI به model (Binding Layout Files)

فایل های layout ای که `data binding` انجام می دهند، با تگ `layout` آغاز شده و المان `data` را به `view` صورت تودرتو در خود دارند و سپس المان آغازین و میزبان `view` را شامل می شود. این تگ `view`

در واقع المانی است که در یک فایل `non-binding`، ریشه یا `root` در آن قرار می گرفت. در زیر نمونه ای از یک فایل `layout` binding layout را مشاهده می کنید:

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android">
    <data>
        <variable name="user" type="com.vogella.android.databinding.User"/>
    </data>
    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent">
        <TextView android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@{user.firstName}"/>
        <TextView android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@{user.lastName}"/>
    </LinearLayout>
</layout>
```

اطلاعات `data-binding` به فایل های `layout` پروژه ی شما اضافه می شود. در طول پروسه `build` پروژه، کلاس های `binding` بر اساس اطلاعات این فایل تولید می شوند. به صورت پیش فرض، اسم کلاس `binding` بر اساس اسم فایل `layout` ساخته می شود. در واقع اسم فایل با سیستم نوشتاری Pascal نگارش شده و سپس پسوند "Binding" به انتهای آن الصاق می شود. برای مثال اگر اسم فایل `layout` شما `activity_main.xml` باشد، اسم کلاس متعاقباً `ActivityMainBinding` خواهد بود. این کلاس تمامی `binding` ها از متغیرهای فایل `layout` (تگ `variable`) که در نمونه فایل فوق مشاهده می کنید) که اشاره گری به `View` های `layout` هستند را شامل می شود و همچنین دقیقاً می داند چگونه مقادیر را به `view` ها متصل (bind) کند. المان `data` که در فایل بالا مشاهده می کنید، داده هایی را توصیف می کند که قرار است به صورت "دو طرفه متصل شوند. Expressions ها یا عبارات داخل فایل های `layout` با دستور نگارشی "@{}" در نوشته `attribute property` می شوند.

در این فایل همچنین تگ `variable` با نام `user` را می بینید که یک متغیر (`property`) تعریف می کند و این متغیر می تواند در `layout` جاری بکار گرفته شود. در واقع این المان اسم یک آبجکت را

مشخص می کند که property های آن در این فایل layout مورد دسترسی قرار می گیرد. برای مثال "بیانگر مقدار فیلد lastName از کلاس User می باشد."

## 2-15-1 event و مدیریت data binding

در اندروید به شما اجازه می دهد تا عبارت هایی بنویسید که event های ارسالی و فعال شده از view ها را مدیریت می کنند (برای مثال می توان به onClick اشاره کرد). اسم attribute های مربوط به event ها بر اساس اسم متدهای فراخوان آن رخداد (listener) مشخص می شود. به طور مثال، View.OnLongClickListener (method) نام onLongClick() دارد و از این رو مرتبط با این event به این صورت نام گذاری می شود: .android:onLongClick

Event ها می توانند به طور مستقیم به متدهای handler متناظر آنها متصل باشند، همان طور که android:onClick می تواند به یک متدهای activity چسبانده شود.

به منظور تخصیص یک event به handler مربوطه، کافی است از یک عبارت اتصال ساده استفاده کنید. به طوری که مقدار اسم همان متدهای باشد که باید با فعال شدن رخداد صدا خورده شود. این عبارت اتصال سپس می تواند listener یا گوش فرادهنده به رخداد کلیک را به view بچسباند.

## 2-15-2 بروز رسانی رابط کاربری با تغییر رخ داده در data model

به منظور اتصال داده از UI می توان از هر آبجکت ساده‌ی جاوایی بهره‌گرفت که در اصطلاح POJO خوانده می شوند. اما متأسفانه تغییر این آبجکت ساده‌ی جاوا به صورت خودکار سبب بروز رسانی UI نمی شود. به منظور فراهم آوردن بروز رسانی UI همراه با تغییر در model، آبجکت حاوی داده‌ها (data object) باید بتوانند بخش UI مربوطه را از تغییراتی که در داده‌ها رخ می دهد، مطلع کنند. در کل سه مکانیزم جهت اطلاع رسانی view ها از تغییرات در model وجود دارد: 1. Observable collections .2 Observable objects .3 Observable fields

برای روش اول، اندروید کلاس BaseObservable را ارائه می دهد که شما می توانید از آن ارث بری کنید. این کلاس که داده‌ها را در خود نگه می دارد، موظف است به محض اینکه مقادیر

property ها تغییر کردند، view ها از این تغییرات با خبر کند. جهت اعطای نمودن این امکان، کافی است دستور @Bindable (annotation) را در بالای تابعی که داده را می خواند درج نموده و مطلع ساختن view از تغییرات را در تابعی که مقدار را تنظیم می کند (setter)، از طریق تابع notifyPropertyChanged(BR.celsius); انجام دهید.

```
package com.vogella.android.databinding;
import android.databinding.BaseObservable;
import android.databinding.Bindable;
import java.util.Observable;
public class TemperatureData extends BaseObservable {
    private String celsius;
    public TemperatureData(String celsius) {
        this.celsius = celsius;
    }
    @Bindable
    public String getCelsius() {
        return celsius;
    }
    public void setCelsius(String celsius) {
        this.celsius = celsius;          1
        notifyPropertyChanged(BR.celsius);
    }
}
```

1. یک getter جهت خواندن داده مربوطه

2. توسط این بخش از کد تمامی listener ها را از تغییرات با خبر می سازید. یک کلاس تولید شده توسط Android studio می باشد.

این listener به محض رخداد هر گونه بروز رسانی و تغییر در داده، فراخوانی می شود و view های متناظر را به تناسب با تغییر رخداده ویرایش می کند. با این مکانیزم برنامه نویس اطمینان حاصل می کند که هرگاه مقادیر model تغییر کردند، بخش UI مربوطه نیز بروز رسانی می شود.

### 15-2-پیاده سازی data binding و نمونه ای از کاربرد آن

همان طور که در بالا نیز گفته شد، فایل های layout ای که از data binding پشتیبانی می کنند با تگ layout آغاز شده و یک المان data را شامل می شود که تعریف کننده داده ای است که قرار است bind یا دوطرفه متصل شوند. پس از المان data، تگ view میزبان و root را داریم که در یک فایل layout ساده و non-binding همان root و عنصر آغازین محسوب می شد. عبارت

های اتصال دو طرفه‌ی داده‌ها (binding expressions) با دستور نگارشی "@{}" در فایل مزبور نوشته می‌شوند.

یک فایل نمونه به صورت زیر نوشته می‌شود.

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android">
    <data>
        <variable name="user" type="com.vogella.android.databinding.User"/>
    </data>
    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent">
        <TextView android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@{user.firstName}"/>
        <TextView android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@{user.lastName}"/>
    </LinearLayout>
</layout>
```

متغیر user در المان data، در واقع یک property می‌کند که می‌تواند داخل این layout مورد استفاده قرار گیرد.

آبجکتی که داده‌ها را در خود کپسوله کرده است (data object) و همراه این layout بکار می‌رود، می‌تواند ظاهری مشابه نمونه‌ی زیر داشته باشد.

```
package com.vogella.android.databinding;
public class User {
    public final String firstName;
    public final String lastName;
    public User(String firstName, String lastName) {
        this.firstName = firstName;
        this.lastName = lastName;
    }
}

package com.vogella.android.databinding;
import android.app.Activity;
import android.databinding.DataBindingUtil;
import android.databinding.ViewDataBinding;
import android.os.Bundle;
import com.vogella.android.databinding.databinding.ActivityMainBinding;
public class MainActivity extends Activity {
    @Override
```

```

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    ActivityMainBinding binding = DataBindingUtil.setContentView(this, R.layout.activity_main);
    User user = new User("Test", "User");
    binding.setUser(user);
}
}

```

## تمرین: استفاده از data binding در پروژه

در طی این پروژه ساده خواهید آموخت چگونه بین المان های رابط کاربری (UI widget) به وسیله data binding تعامل برقرار نمایید. ابتدا یک اپلیکیشن جدید اندرویدی با اسم پکیج com.vogella.android.databinding ایجاد نمایید.

### ۱۵-۲- افزودن کتابخانه data binding به پروژه

فایل app/build.gradle را باز نموده و با افزودن تکه کد زیر به بخش مربوطه در این فایل، قابلیت data binding را فعال نمایید. توجه کنید که باید build file مناسب را انتخاب نمایید (آن فایلی که نود application را دارد).

```

apply plugin: 'com.android.application'
android {
    dataBinding {
        enabled = true
    }
    .... [REST AS BEFORE...]
}

```

### ۱۵-۳- ساخت کلاس جهت مطابع ساختن view از تغییرات در model

کلاس های زیر را تعریف نمایید.

```

package com.vogella.android.databinding;
import android.databinding.BaseObservable;
import android.databinding.Bindable;
import android.databinding.ObservableField;
import java.util.Observable;
public class TemperatureData extends BaseObservable{
    private String celsius;
    public TemperatureData(String celsius) {
        this.celsius = celsius;
    }
    private String fahrenheit;
}

```

```

@Bindable
public String getCelsius() {
    return celsius;
}
public void setCelsius(String celsius) {
    this.celsius = celsius;
    notifyPropertyChanged(BR.celsius);
}
}

package com.vogella.android.databinding;
public interface MainActivityContract {
    public interface Presenter {
        void onShowData(TemperatureData temperatureData);
    }
    public interface View {
        void showData(TemperatureData temperatureData);
    }
}

package com.vogella.android.databinding;
import android.util.Log;
import android.view.View;
import android.widget.Toast;
public class MainActivityPresenter {
    private MainActivityContract.View view;
    public MainActivityPresenter(MainActivityContract.View view) {
        this.view = view;
    }
    public void onShowData(TemperatureData temperatureData) {
        view.showData(temperatureData);
    }
}

```

## 15-2- تنظیم و ویرایش فایل ها برای استفاده از قابلیت

فایل layout را به صورت زیر ویرایش نمایید.

```

<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android">
    <data>
        <variable
            name="temp"
            type="com.vogella.android.databinding.TemperatureData" />
        <variable
            name="presenter"
            type="com.vogella.android.databinding.MainActivityPresenter"/>
    </data>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"

```

```

    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin">
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@={temp.celsius}"
            android:id="@+id/textView" />
        <EditText
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="8dp"
            android:text="@={temp.celsius}" />
        <Button
            android:text="Show data model"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:onClick="@{() -> presenter.onShowData(temp)}"
            android:id="@+id/button" />
    
```

**توجه:** ممکن است تعدادی پیغام هشدار در ویرایشگر مشاهده نمایید. برای مثال ممکن است پیغامی مبنی بر اینکه رشته های hard-code شده در پروژه استفاده شده است دریافت نمایید. در بخش های آتی برای شما شرح می دهیم چگونه این پیغام ها را برطرف نمایید.

```

package com.vogella.android.databinding;
import android.app.Activity;
import android.databinding.DataBindingUtil;
import android.os.Bundle;
import android.widget.Toast;
import com.vogella.android.databinding.databinding.ActivityMainBinding;
public class MainActivity extends Activity implements MainActivityContract.View {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        ActivityMainBinding binding = DataBindingUtil.setContentView(this, R.layout.activity_main);
        MainActivityPresenter mainActivityPresenter = new MainActivityPresenter(this);
        TemperatureData temperatureData = new TemperatureData("10");
        binding.setTemp(temperatureData);
        binding.setPresenter(mainActivityPresenter);
    }
    @Override
    public void showData(TemperatureData temperatureData) {
        String celsius = temperatureData.getCelsius();
        Toast.makeText(this, celsius, Toast.LENGTH_SHORT).show();
    }
}

```

کد کلاس `MainActivity` را برای فعال سازی `data binding` تنظیم نمایید.

## تست اپلیکیشن

می توانید اپلیکیشن خود را در محیط شبیه ساز راه اندازی نمایید. با کلیک بر روی دکمه، یک `popup` حامل داده های مرتبط به نمایش در می آید. با تایپ کردن در `EditText`، می بینید که مقادیر المان `TextView` به صورت خودکار بروز رسانی می شود.

بخش ششم :

## 16-2-استفاده از کلاس `ListView` در اندروید

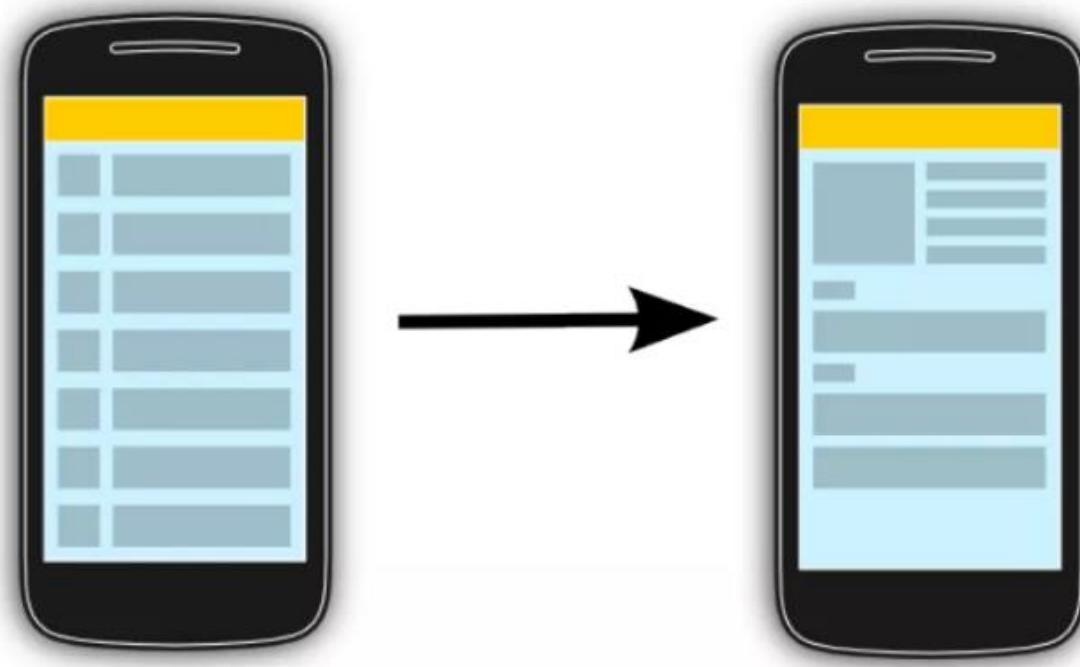
پیاده سازی `list` در اندروید با کلاس `ListView`

مقاله‌ی حاضر به شما آموزش می دهد چگونه `ListView` را با `activity` و `fragment` در اندروید بکار ببرید.

نمایش مجموعه ای سازماندهی شده از المان ها در اندروید یک الگوی بسیار معمول در اپلیکیشن های تحت موبایل است. کاربر رو به رو خود لیستی از آیتم ها را می بیند و می تواند به راحتی داخل آن پیمایش کند. چنین `activity` در تصویر زیر به نمایش گذاشته شده است.



به طور معمول، کاربر از طریق نوار ابزار با لیست تعامل برقرار می کند. به طور مثال، یک دکمه در نوار ابزار که لیست را بروز رسانی نموده و آیتم هایی را از آن حذف یا به آن اضافه می کند. آیتم های فردی لیست را می توان انتخاب نمود که به دنبال این انتخاب نوار ابزار بروزآوری شده یا ممکن است یک صفحه *i* جدید با جزئیات فراوان به نمایش در آید. در زیر می بینید که با انتخاب یک آیتم از لیست، دیگری راه اندازی شده و برای کاربر به نمایش در می آید.



## 17-2-Android و ListView رابط کاربری/المان

### 17-1 View هایی برای مدیریت لیست ها

چارچوب نرم افزاری (framework) اندروید برای پیاده سازی لیست های ساده، دو کلاس `ListView` و `ExpandableListView` را ارائه می دهد. با بهره گیری از این دو کلاس، برنامه نویس قادر خواهد بود به راحتی لیست های قابل پیمایش با نوار اسکرول را پیاده سازی کند.

کلاس `ExpandableListView` علاوه بر سازمان دهنده المان ها، آن ها را در دو سطح یا گروه متمایز قرار می دهد، به طوری که کاربر می تواند با کلیک بر روی هر یک المان های فرزند آن دو را به صورت جداگانه مشاهده کند.

## 2-17-2-نوع داده ای آیتم های لیست

آیتم هایی که در لیست به نمایش در می آیند، می توانند یک آبجکت جاوا از هر نوعی باشد. کلاس adapter داده های مربوطه را از `data object` (کلاسی که داده ها در آن کپسوله شده اند) استخراج کرده و داده های واکنشی شده را به `view` های متناظر در `ListView` تخصیص می دهد.

این آیتم ها به طور معمول همان `data model` لیست خوانده می شوند. کلاس `adapter` این داده ها را به عنوان ورودی دریافت می کند. در حقیقت `adapter` یک پل ارتباطی است که داده ها را از منبع داده (مثلا آرایه) خوانده و آن ها را به `Adapter view` تحویل می دهد. این داده ها نهایتا در اختیار کامپوننت های UI قرار داده شده و در نمایشگر ارائه می شوند.

## 2-17-3 Adapter ها

در واقع `data model` اپلیکیشن را مدیریت می کند و آن را با توجه به المان های فردی `widget` بروز رسانی کرده و تطبیق می دهد. کلاس `adapter` خود از کلاس `BaseAdapter` مشتق می شود.

هر خط که در `widget` یا المان رابط کاربری اطلاعاتی را نشان می دهد، خود از یک `layout` تشکیل شده و می تواند بر اساس نیاز پیچیده یا ساده باشد. ظاهر کلی سطرها در لیست اغلب به این صورت است که در سمت چپ یک عکس قابل مشاهده است و در وسط دو خط متن مانند تصویر زیر را شامل می شود.



فایل `layout` متناظر این سطر از لیست می تواند به صورت زیر باشد.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="?android:attr/listPreferredItemHeight"
    android:padding="6dp" >
```

```

<ImageView
    android:id="@+id/icon"
    android:layout_width="wrap_content"
    android:layout_height="fill_parent"
    android:layout_alignParentBottom="true"
    android:layout_alignParentTop="true"
    android:layout_marginRight="6dip"
    android:contentDescription="TODO"
    android:src="@drawable/ic_launcher" />
<TextView
    android:id="@+id/secondLine"
    android:layout_width="fill_parent"
    android:layout_height="26dip"
    android:layout_alignParentBottom="true"
    android:layout_alignParentRight="true"
    android:layout_toRightOf="@+id/icon"
    android:ellipsize="marquee"
    android:singleLine="true"
    android:text="Description"
    android:textSize="12sp" />
<TextView
    android:id="@+id/firstLine"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_above="@+id/secondLine"
    android:layout_alignParentRight="true"
    android:layout_alignParentTop="true"
    android:layout_alignWithParentIfMissing="true"
    android:layout_toRightOf="@+id/icon"
    android:gravity="center_vertical"
    android:text="Example application"
    android:textSize="16sp" />
</RelativeLayout>

```

کلاس adapter به ازای هر سطر در لیست، layout مربوطه را inflate می کند (آن را در حافظه بارگذاری و آماده ی نمایش در UI می نماید). در واقع با استفاده از متدهای getView() هر layout را واکشی و سپس داده های view را به view های مرتبط در سطر میزبان متصل می کند.

از طریق متدهای setAdapter() که به آبجکت ListView الحاق شده، به Adapter متصل می شود.

**توجه:** این تنها ListView نیست که از Adapter استفاده می کند، بلکه سایر view هایی که از ارث بری دارند، نظیر StackView و Gallery و Spinner نیز برای واکشی داده های خود با این کلاس همکاری دارند.

## 4-17-2-فیلتر و مرتب سازی داده ها

فیلتر کردن و مرتب سازی داده ها بر عهده `adapter` می باشد. منطق مربوطه را می بایست در بدن `adapter` اختصاصی خود (که از کلاس پدر ارث بری کردید) پیاده سازی نمایید.

## 4-17-3-بروز رسانی داده ها در adapter

متدهای `notifyDataSetChanged()` در کلاس `adapter` زمانی فراخوانی می شود که داده ها تغییر کرده باشند یا داده های جدید در دسترس قرار گرفته باشد.

متدهای `notifyDataSetInvalidated()` زمانی صدا خورده می شود که داده ها دیگر در دسترس نباشند.

## 4-17-4-تعريف listener برای آگاهی از تغییرات

برای واکنش نشان دادن به تغییراتی که در لیست رخ می دهد (برای مثال انتخابی که کاربر می کند)، لازم است یک `ListView` به `OnItemClickListener` خود الحاق کنید.

```
listView.setOnItemClickListener(new OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view,
            int position, long id) {
        Toast.makeText(getApplicationContext(),
                "Click ListItem Number " + position, Toast.LENGTH_LONG)
                .show();
    }
});
```

## 18-2 پیش فرض Adapter

## 18-2-کلاس های adapter پیش فرض محیط (platform) اندروید

اندروید تعدادی adapter با پیاده سازی پیش فرض ارائه می دهد که مهم ترین و پرکاربردترین آن ها به شرح زیر می باشد: CursorAdapter.ArrayAdapter و ArrayAdapter که زمانی استفاده می شود که منبع داده ای آرایه باشد و دیگری java.util.List. SimpleCursorAdapter که می تواند به هنگام کار با داده های مستقر در دیتابیس مورد استفاده قرار گیرد (منبع داده ای یک cursor برای ردیف های دیتابیس باشد).

## 18-2-استفاده از ArrayAdapter

کلاس ArrayAdapter می تواند یک لیست یا آرایه ای از آبجکت های جاوا را به عنوان ورودی دریافت کند. هر آبجکت جاوایی به یک سطر نگاشت می شود. در واقع متد ()toString() را به ازای هر المان در آرایه صدا می زند و یک شی view می سازد و آن را در TextView قرار می دهد.

شما می توانید ID یا شناسه ی منحصر بفرد view را در تابع سازنده (Constructor) کلاس ArrayAdapter تعریف کنید. در غیر این صورت شناسه ی android.R.id.text1 به صورت پیش فرض مورد استفاده قرار می گیرد.

کلاس ArrayAdapter به شما این امکان را می دهد تمامی المان های موجود در منبع داده (data structure) را با فراخوانی متد ()clear حذف نمایید. سپس شما می توانید به وسیله ی متد ()add() المان های جدید و به وسیله ی ()addAll() یک Collection جدید اضافه نمایید.

این امکان نیز برای توسعه دهنده وجود دارد که محتوای منبع داده (برای مثال آرایه) را ویرایش نموده و سپس با فراخوانی متد ()notifyDataSetChanged() در.adapter این آبجکت را از تغییرات در داده ها آگاه نمایید.

**توجه:** اگر می خواهید داده ها در adapter خود تغییر دهید، در آن صورت منبع داده ی اصلی ArrayList (underlying data structure) نیز باید این عملیات را پشتیبانی کند. برای مثال، کلاس این امکان را به شما می دهد ولی این امر درمورد آرایه صادق نیست.

## 3-18-2-نمونه‌ی استفاده از ArrayAdapter با ListView

کد XML زیر، یک قطعه از فایل layout به نام activity\_listviewexampleactivity.xml به نمایش می‌گذارد که کامپوننت رابط کاربری ListView را شامل می‌شود.

```
<ListView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/listview"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

مثال زیر کاربرد آبجکت ListView در یک activity را به نمایش می‌گذارد. این مثال برای چیدمان و تنظیم ظاهر سطرها (row layout) از یک layout پیش فرض و آماده‌ی محیط اندروید (platform) بهره می‌گیرد. علاوه بر آن مثال حاضر حذف آیتم‌ها از لیست را همراه با انیمیشن به نمایش می‌گذارد.

```
package com.vogella.android.listview.withanimation;
public class ListViewExampleActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_listviewexampleactivity);
        final ListView listview = (ListView) findViewById(R.id.listview);
        String[] values = new String[] { "Android", "iPhone", "WindowsMobile",
            "Blackberry", "WebOS", "Ubuntu", "Windows7", "Max OS X",
            "Linux", "OS/2", "Ubuntu", "Windows7", "Max OS X", "Linux",
            "OS/2", "Ubuntu", "Windows7", "Max OS X", "Linux", "OS/2",
            "Android", "iPhone", "WindowsMobile" };
        final ArrayList<String> list = new ArrayList<String>();
        for (int i = 0; i < values.length; ++i) {
            list.add(values[i]);
        }
        final StableArrayAdapter adapter = new StableArrayAdapter(this,
            android.R.layout.simple_list_item_1, list);
        listview.setAdapter(adapter);
        listview.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, final View view,
                int position, long id) {
                final String item = (String) parent.getItemAtPosition(position);
                view.animate().setDuration(2000).alpha(0)
                    .withEndAction(new Runnable() {
                        @Override
                        public void run() {
                            list.remove(item);
                        }
                    });
            }
        });
    }
}
```

```

        adapter.notifyDataSetChanged();
        view.setAlpha(1);
    }
}
};

private class StableArrayAdapter extends ArrayAdapter<String> {

    HashMap<String, Integer> mIdMap = new HashMap<String, Integer>();
    public StableArrayAdapter(Context context, int textViewResourceId,
        List<String> objects) {
        super(context, textViewResourceId, objects);
        for (int i = 0; i < objects.size(); ++i) {
            mIdMap.put(objects.get(i), i);
        }
    }
    @Override
    public long getItemId(int position) {
        String item = getItem(position);
        return mIdMap.get(item);
    }
    @Override
    public boolean hasStableIds() {
        return true;
    }
}
}

```

## 19-2-پیاده سازی adapter های اختصاصی

### 19-2-1-ساخت یک adapter اختصاصی

برای مدیریت تخصیص داده و پشتیبانی از چندین view، می بایست خود یک کلاس adapter با پیاده سازی اختصاصی تعریف نمایید.

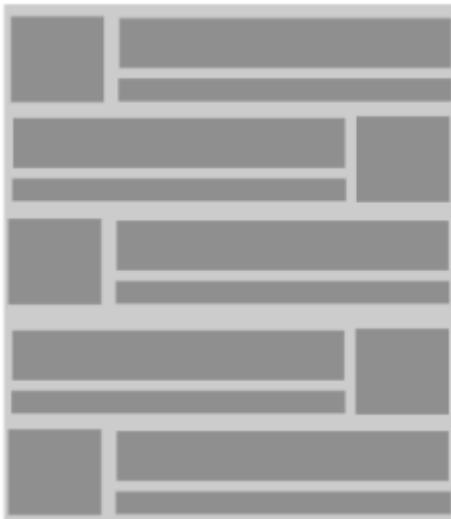
برای این منظور ابتدا لازم است یک کلاس پیاده سازی شده‌ی adapter را به ارث ببرید و یا به طور مستقیم از کلاس BaseAdapter یک کلاس فرزند ایجاد کرده و بدنه‌ی آن را خود پیاده سازی کنید.

**توجه:** اغلب توسعه دهنده از کلاس ArrayAdapter ارث بری کرده و یک adapter اختصاصی پیاده سازی می‌کند. این کار به مراتب ساده‌تر از ایجاد کلاس فرزند و ارث بری مستقیم از BaseAdapter است.

## 2-19-2-آماده سازی سطر از لیست

Adapter می‌باشد به ازای هر سطر از لیست، یک layout ایجاد کند. آبجکت (instance) ListView متد () getview در لیست، در سطح کلاس Adapter صدا می‌زند. به واسطه‌ی این layout متد adapter مربوط به سطر را ایجاد کرده و داده‌ها را به view‌های مربوطه در نگاشت می‌کند.

این root معمولاً یک ViewGroup (یک layout manager) است که خود میزبان چندین view دیگر می‌باشد. به عنوان مثال می‌توان از TextView و ImageView نام برد. تصویر زیر یک لیست را با layout های متفاوت برای سطرهای زوج و فرد نشان می‌دهد.



متد () getView() فایل XML را خوانده و آن را به listview تبدیل می کند.

حال با استفاده از متد () getView() فایل layout مبتنی بر XML خود را بارگذاری یا inflate کرده و سپس محتوای تک تک view های سطر مورد نظر را بر اساس آبجکت جاوا تنظیم می کنید (آن ها را با داده هایی از آبجکت جاوا پر می کنید). به منظور inflate کردن فایل layout مبتنی بر XML خود، می توانید از سرویس LayoutInflator استفاده نمایید.

**توجه:** این سرویس که layout ها را خوانده و در حافظه بارگذاری می کند (inflate)، به راحتی به واسطه ای فراخوانی متد () getLayoutInflator در سطح کلاس activity و یا فراخوانی متد context.getSystemService(Context.LAYOUT\_INFLATER\_SERVICE) به راحتی قابل دسترسی می باشد.

پس از اینکه adapter فایل layout مربوطه را در حافظه بارگذاری و به اصطلاح inflate کرد، view های متناظر در layout را پیدا می کند و متعاقبا آن ها را با داده پر می نماید. همان طور که قبل نیز به آن اشاره شده، جهت پیدا کردن المان های فردی در layout، کافی است متد findViewById() را بر روی view (top level view) میزبان که view مورد نظر در آن تعریف شده است) فراخوانی نمایید.

## نمونه ای از پیاده سازی یک adapter اختصاصی و custom

کد زیر بدنه ای یک کلاس adapter با پیاده سازی اختصاصی را نشان می دهد. این آبجکت فرض می گیرد شما دو فایل تصویری png (ok.png و no.png) در یکی از پوشش های directory]]res/drawable خود دارید. کد نوشته شده در این adapter، فایل layout XML را در حافظه بارگذاری (inflate) کرده، view های مربوطه را در layout پیدا می کند و سپس محتوای آن ها را بر اساس داده های ورودی مقداردهی و تنظیم می نماید.

```
package de.vogella.android.listactivity;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
```

```

import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
import android.widget.TextView;

public class MySimpleArrayAdapter extends ArrayAdapter<String> {
    private final Context context;
    private final String[] values;
    public MySimpleArrayAdapter(Context context, String[] values) {
        super(context, -1, values);
        this.context = context;
        this.values = values;
    }
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        LayoutInflator inflater = (LayoutInflator) context
                .getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        View rowView = inflater.inflate(R.layout.rowlayout, parent, false);
        TextView textView = (TextView) rowView.findViewById(R.id.label);
        ImageView imageView = (ImageView) rowView.findViewById(R.id.icon);
        textView.setText(values[position]);
        // change the icon for Windows and iPhone
        String s = values[position];
        if (s.startsWith("iPhone")) {
            imageView.setImageResource(R.drawable.no);
        } else {
            imageView.setImageResource(R.drawable.ok);
        }
        return rowView;
    }
}

```

### 19-2-بروز رسانی adapter از طریق کلاس data model

سطر مورد نظر می تواند view هایی داشته باشد که با data model (آبجکتی که داده ها را کپسوله می کند) از طریق کلاس adapter تعامل دارند. به طور مثال، شما می توانید یک Checkbox در layout مربوط به سطر داشته باشید و هر زمان که این Checkbox تیک دار و به اصطلاح انتخاب شد، داده های موجود در آبجکت model نیز همگام با آن تغییر کرده و بروز رسانی شوند.

## ListFragment و ListActivity-20-2

## 20-2-ظرف آماده و پیش فرض برای استفاده از ListView

اندروید با ارائه‌ی fragment‌ها و کلاس‌های activity ویژه، پیاده‌سازی و مدیریت لیست را برای توسعه دهنده بسیار آسان می‌کند.

برای ایجاد لیست در activity و fragment، اندروید به ترتیب کلاس‌های ListActivity و ListFragment را ارائه می‌دهد.

لازم به ذکر است که برای این المان‌ها نیازی به تخصیص layout نیست چرا که activity یا fragment مورد نظر خود به صورت پیش فرض خود یک ListView آماده دارد.

(override) و ListFragment به برنامه نویس این اجازه را می‌دهند تا با بازنویسی ListActivity پیاده‌سازی متدهای onListItemClick() انتخاب آیتم‌ها در لیست را مدیریت کند.

هر دو کلاس همچنین این اختیار به برنامه نویس می‌دهند تا adapter را به وسیله‌ی متدهای SetListAdapter() به ListView پیش فرض متصل کند.

نمونه کد زیر یک پیاده‌سازی ساده از ListFragment را نشان می‌دهد.

```
package de.vogella.android.fragments;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.app.ListFragment;
public class MyListFragment extends ListFragment {
    @Override
    public void onActivityCreated(Bundle savedInstanceState) {
        super.onActivityCreated(savedInstanceState);
        String[] values = new String[] { "Android", "iPhone", "WindowsMobile",
            "Blackberry", "WebOS", "Ubuntu", "Windows7", "Max OS X",
            "Linux", "OS/2" };
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(getActivity(),
            android.R.layout.simple_list_item_1, values);
        setListAdapter(adapter);
    }
    @Override
    public void onListItemClick(ListView l, View v, int position, long id) {
        // TODO implement some logic
    }
}
```

```

    }
}

```

نمونه کد بعدی که مشاهده می کنید، استفاده از `ListActivity` را به نمایش می گذارد.

```

package de.vogella.android.listactivity;
import android.app.ListActivity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
public class MyListActivity extends ListActivity {
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        String[] values = new String[] { "Android", "iPhone", "WindowsMobile",
            "Blackberry", "WebOS", "Ubuntu", "Windows7", "Max OS X",
            "Linux", "OS/2" };
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
            android.R.layout.simple_list_item_1, values);
        setListAdapter(adapter);
    }
}

```

## 2-20-2 layout و ListActivity

شما می توانید برای هر دو کلاس `ListFragment` و `ListActivity` فایل `layout` اختصاصی داشته باشید. در این مثال، `fragment` یا `activity` در `layout` ارائه شده به دنبال `ListView` ای که مقدار آن به صورت پیش فرض بر روی `@android:id/list` تنظیم شده است، می گردد. تکه کد زیر این کاربرد را به نمایش می گذارد.

```

<ListView
    android:id="@+id/list"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >
</ListView>

```

**توجه:** در صورتی که از این ID استفاده نکرده یا `ListView` را در `layout` خود اضافه نکنید، اپلیکیشن به هنگام نمایش `activity` یا `fragment` مربوطه از کار می افتد.

## 2-20-3 انتخاب یک view برای placeholder

می توانید از یک `view` با شاسنه `@+id/empty` در `layout` خود استفاده کنید. چنانچه تهی باشد، در آن صورت `activity` و `fragment` مربوطه این `view` را به صورت خودکار

نمایش می دهند و در غیر این صورت آن را به طور کلی مخفی می کند. در چنین view ای شما می توانید یک پیغام خطأ را به نمایش بگذارید.

## تمرین: استفاده از ListView و ListActivity

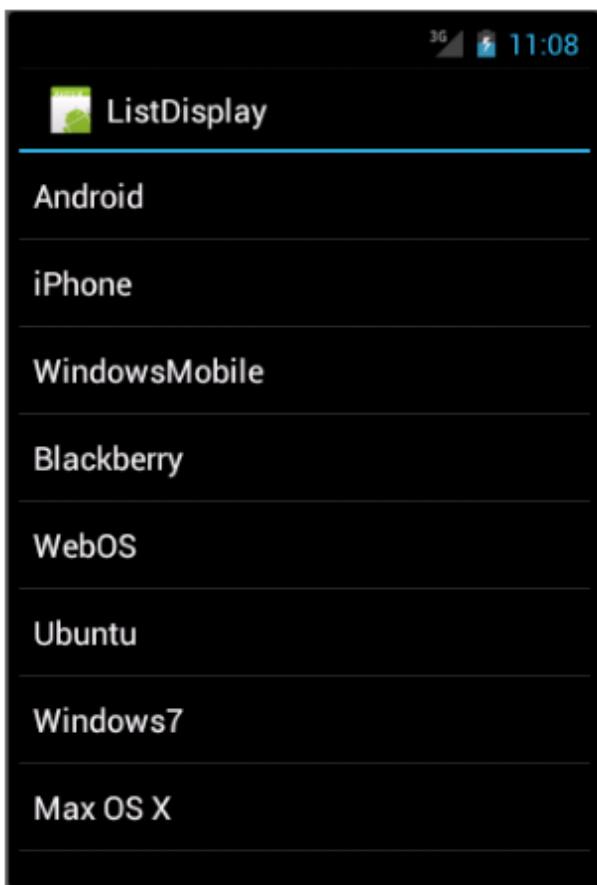
پروژه‌ی زیر نحوه‌ی استفاده از یک ListActivity در ListView را نمایش می‌دهد. در این پروژه شما از کلاس از پیش تعریف شده‌ی ArrayAdapter و یک آماده‌ی اندروید برای سطرا استفاده می‌کنید.

یک پروژه‌ی جدید اندروید به نام de.vogella.android.listactivity و یک کلاس به نام MyListActivity ایجاد کنید.

کلاس MyListActivity را بر اساس نمونه کد زیر ویرایش نمایید. همان طور که می‌بینید متده setContentView() استفاده نشده است.

```
package de.vogella.android.listactivity;
import android.app.ListActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;
public class MyListActivity extends ListActivity {
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        String[] values = new String[] { "Android", "iPhone", "WindowsMobile",
            "Blackberry", "WebOS", "Ubuntu", "Windows7", "Max OS X",
            "Linux", "OS/2" };
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
            android.R.layout.simple_list_item_1, values);
        setContentView(adapter);
    }
    @Override
    protected void onListItemClick(ListView l, View v, int position, long id) {
        String item = (String) getListAdapter().getItem(position);
        Toast.makeText(this, item + " selected", Toast.LENGTH_LONG).show();
    }
}
```

}



## تمرين: ListActivity با layout اختصاصی

در مثال حاضر، فایل layout مربوط به سطرهای را خود به صورت اختصاصی تعریف کرده و آن را در adapter بکار می‌برید.

ابتدا فایل rowlayout.xml را در پوشه res/layout ایجاد نمایید.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" >
    <ImageView
        android:id="@+id/icon"
```

```

    android:layout_width="22px"
    android:layout_height="22px"
    android:layout_marginLeft="4px"
    android:layout_marginRight="10px"
    android:layout_marginTop="4px"
    android:src="@drawable/ic_launcher" >
</ImageView>
<TextView
    android:id="@+id/label"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@+id/label"
    android:textSize="20px" >
</TextView>
</LinearLayout>

```

Activity خود را جهت استفاده از layout جدید ویرایش نمایید.

```

package de.vogella.android.listactivity;
import android.app.ListActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;
public class MyListActivity extends ListActivity {
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        String[] values = new String[] { "Android", "iPhone", "WindowsMobile",
            "Blackberry", "WebOS", "Ubuntu", "Windows7", "Max OS X",
            "Linux", "OS/2" };
        // use your custom layout
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
            R.layout.rowlayout, R.id.label, values);
        setListAdapter(adapter);
    }
    @Override
    protected void onListItemClick(ListView l, View v, int position, long id) {
        String item = (String) getListAdapter().getItem(position);
        Toast.makeText(this, item + " selected", Toast.LENGTH_LONG).show();
    }
}

```



## آموزش: پیاده سازی adapter اختصاصی

برنامه‌ی زیر از دو فایل تصویری "ok.png" و "no.png" استفاده می‌کند. این دو فایل در پوشه‌ی "res/drawable-mdpi" قرار دارند. شما با استفاده از آیکون‌های خود را ایجاد کنید. در صورتی که آیکون دلخواه را پیدا نکردید، می‌توانید "icon.png" را کپی کرده و با استفاده از یک برنامه‌ی گرافیکی آن را کمی تغییر دهید و دومین آیکون را هم به همین ترتیب ایجاد نمایید.

کلاس `MySimpleArrayAdapter` را ایجاد کنید. این کلاس `adapter` و پل ارتباطی بین `model` و `UI` خواهد بود.

```
package de.vogella.android.listactivity;
import android.content.Context;
import android.view.LayoutInflater;
```

```

import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
import android.widget.TextView;
public class MySimpleArrayAdapter extends ArrayAdapter<String> {
    private final Context context;
    private final String[] values;
    public MySimpleArrayAdapter(Context context, String[] values) {
        super(context, R.layout.rowlayout, values);
        this.context = context;
        this.values = values;
    }
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        LayoutInflater inflater = (LayoutInflater) context
            .getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        View rowView = inflater.inflate(R.layout.rowlayout, parent, false);
        TextView textView = (TextView) rowView.findViewById(R.id.label);
        ImageView imageView = (ImageView) rowView.findViewById(R.id.icon);
        textView.setText(values[position]);
        // Change the icon for Windows and iPhone
        String s = values[position];
        if (s.startsWith("Windows7") || s.startsWith("iPhone")
            || s.startsWith("Solaris")) {
            imageView.setImageResource(R.drawable.no);
        } else {
            imageView.setImageResource(R.drawable.ok);
        }
        return rowView;
    }
}

```

برای استفاده از این adapter لازم است activity را به صورت زیر ویرایش نمایید.

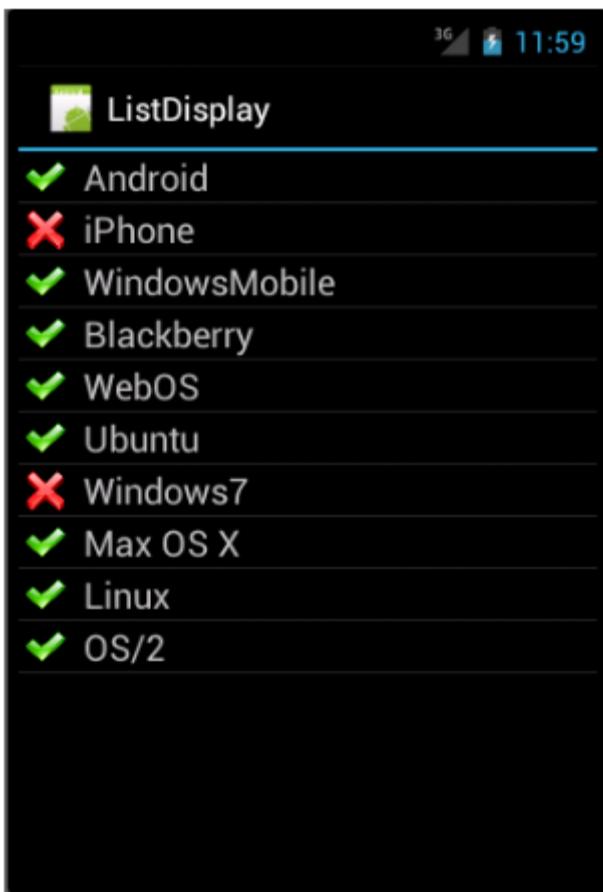
To use this adapter, change the *activity* to the following.

```

package de.vogella.android.listactivity;
import android.app.ListActivity;
import android.os.Bundle;
public class MyListActivity extends ListActivity {
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        String[] values = new String[] { "Android", "iPhone", "WindowsMobile",
            "Blackberry", "WebOS", "Ubuntu", "Windows7", "Max OS X",
            "Linux", "OS/2" };
        MySimpleArrayAdapter adapter = new MySimpleArrayAdapter(this, values);
        setListAdapter(adapter);
    }
}

```

با اجرای این برنامه لیستی از آیتم ها را مشاهده می کنید که کنار برخی آیکون ضرب در و کنار برخی دیگر تیک دیده می شود.



## ListView-21-2 و مبحث کارایی و سرعت اپلیکیشن

کارایی و سرعت در اندروید از اهمیت بسیار بالایی برخوردار است و کاربران انتظار واکنش سریع و سرعت بالای پاسخگویی از اپلیکیشن را دارند. از لحاظ قدرت سخت افزاری و منابع موجود، یک دستگاه اندروید به هیچ وجه با کامپیوترهای دسکتاپ قابل مقایسه نیست.

در این بخش شرح خواهیم داد چگونه adapter لیست خود را به صورت بهینه پیاده سازی نمایید و عملیات غیر ضروری را جهت افزایش سرعت اپلیکیشن و صرفه جویی در مصرف منابع سیستم

کاهش دهید. گفتنی است که adapter های پیش فرض اندروید همچون ArrayAdapter طوری طراحی شده اند که از لحاظ کارایی کاملاً بهینه هستند.

## 21-2-عملیات سنگین و زمان بر

تمامی view هایی که از فایل layout XML خوانده و inflate می شوند، در نهایت به آبجکت های Java تبدیل می شوند. این دو فرایند هر دو علاوه بر زمان بر بودن، حافظه‌ی قابل توجهی را به خود اختصاص می دهند.

علاوه، استفاده از متد ()findViewById بسیار زمان بر است، هرچند به اندازه‌ی inflate کردن فایل های XML سنگین نیست.

## 21-2-جلوگیری از inflation فایل های layout و ایجاد آبجکت های جاوا

### جهت بهینه سازی برنامه

یک آبجکت ListView معمولاً اطلاعات بیشتری نسبت به سطرهای قابل مشاهده در صفحه دربردارد. چنانچه کاربر با اسکرول داخل لیست پیمایش کند، آن هنگام سطرها و view های مربوطه‌ی هریک از ناحیه‌ی قابل مشاهده برای کاربر خارج می شوند. آبجکت های جاوا که این سطرها را به نمایش می گذارند و در واقع کد پشت view ها هستند، قابل بازیافت و استفاده‌ی مجدد برای سطرهایی می باشند که با پیمایش کاربر، در صفحه به نمایش در می آیند.

اگر اندروید تشخیص دهد که سطری دیگر در صفحه جاری قابل مشاهده نیست، به متد ()getView از کلاس adapter اجازه می دهد تا view مرتبط را از طریق پارامتر convertView بازیافت کند (reuse).

کلاس adapter می تواند داده های جدید را به view های موجود در سلسله مراتب view ها که با پارامتر convertView در دسترس قرار گرفته، متصل کند. این امر مانع از inflate شدن فایل XML و ایجاد آبجکت های جدید Java می شود.

چنانچه امکان بازیافت یک سطر وجود نداشت (reuse)، سیستم اندروید باید مقدار null را به پارامتر ارسال کند. در واقع، این کد موجود در بدنه‌ی adapter است که باید مکانیزمی برای مدیریت این سناریو پیاده سازی کند.

### 21-2-استفاده از الگوی توسعه‌ی View holder جهت بهینه سازی

پیاده سازی الگو توسعه‌ی ViewHolder این امکان را فراهم می‌کند تا از فراخوانی متعدد findViewById() در کلاس adapter جلوگیری شود.

معمولًا یک کلاس درون ساخته static در قابل متغیر به view های مربوطه (که متناظر آن ها در فایل layout تعریف شده) در خود نگهداری می‌کند. این اشاره گرها به وسیله‌ی متد () و به صورت یک تگ (view setTag()) به متناظر در سطر مورد نظر متصل می‌شوند.

چنانچه آبجکت convertView را دریافت کردیم، آنگاه می‌توانیم نمونه‌ای از ViewHolder را با متد getTag() بازیابی کرده و attribute های جدید را به وسیله‌ی اشاره گر (متغیر متناظر در view به ViewHolder تخصیص دهیم).

اگرچه این روش کمی پیچیده به نظر می‌رسد، اما حدوداً ۱۵٪ سریع‌تر از فراخوانی متعدد findViewById() و پیدا کردن تک تک view ها در فایل XML می‌باشد.

### مثال

کد زیر نمونه‌ای بهینه سازی شده از یک adapter را نشان می‌دهد که از الگوی توسعه‌ی view holder استفاده نموده و با بازیافت view های جاری در استفاده از منابع و حافظه صرفه جویی می‌کند.

```

package de.vogella.android.listactivity;
import android.app.Activity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
import android.widget.TextView;
public class MyPerformanceArrayAdapter extends ArrayAdapter<String> {
    private final Activity context;
    private final String[] names;
    static class ViewHolder {
        public TextView text;
        public ImageView image;
    }
    public MyPerformanceArrayAdapter(Activity context, String[] names) {
        super(context, R.layout.rowlayout, names);
        this.context = context;
        this.names = names;
    }
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        View rowView = convertView;
        // reuse views
        if (rowView == null) {
            LayoutInflater inflater = context.getLayoutInflater();
            rowView = inflater.inflate(R.layout.rowlayout, null);
            // configure view holder
            ViewHolder viewHolder = new ViewHolder();
            viewHolder.text = (TextView) rowView.findViewById(R.id.TextView01);
            viewHolder.image = (ImageView) rowView
                .findViewById(R.id.ImageView01);
            rowView.setTag(viewHolder);
        }
        // fill data
        ViewHolder holder = (ViewHolder) rowView.getTag();
        String s = names[position];
        holder.text.setText(s);
        if (s.startsWith("Windows7") || s.startsWith("iPhone")
            || s.startsWith("Solaris")) {
            holder.image.setImageResource(R.drawable.no);
        } else {
            holder.image.setImageResource(R.drawable.ok);
        }
        return rowView;
    }
}

```

## 2-22- ذخیره سازی انتخاب یک view (تعیین وضعیت انتخاب با `setChoiceMode()`)

به صورت پیش فرض، `ListView` هیچ حالت انتخاب پیش فرض و فعالی ندارد. شما می توانید این حالت را با فراخوانی متد `setChoiceMode()` فعال نمایید. برای اینکه کاربر بتواند چندین آیتم را انتخاب کند، می بایست پارامتر `ListView.CHOICE_MODE_MULTIPLE` را به متد مذکور ارسال کنید و اگر می خواهید امکان انتخاب یک آیتم را بیشتر نداشته باشد، مقدار `ListView.CHOICE_MODE_SINGLE` را به عنوان آرگومان به متد مورد نظر پاس می دهید.

جهت بازیابی آیتم های انتخاب شده ای `ListView`، اگر یک آیتم انتخاب شده باشد، متد `getCheckedItemPosition()` را فراخوانی می کنید و اگر چندین آیتم انتخاب شده باشد، آیتم `listView.getCheckedItemPositions()` می توانید با استفاده از متد `getCheckedItemIds()` آیتم های انتخاب شده را بازگردانی نمایید.

اندروید از قبل برای تنظیم ظاهر لیست هایی که چندین آیتم در آن انتخاب شده باشد، یک `layout` آماده ارائه می دهد: `android.R.layout.simple_list_item_multiple_choice`. این فایل دربردارنده ی `view` کاملا تنظیم شده `CheckedTextView` می باشد.

در `activity` های زیر نحوه ای استفاده از این حالت های انتخاب به نمایش گذاشته شده است. لازم به ذکر است که در صورت استفاده از این حالت ها، `ListView` مقادیر انتخابی را ذخیره می کند. اما اطلاعات مذکور به صورت دائمی در `data model` ذخیره و ماندگار نمی شوند.

```
package com.vogella.android.listview.selection.multi;
import android.app.ListActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;
import com.vogella.android.listview.selection.R;
public class MainActivity extends ListActivity {
```

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    String[] values = new String[] { "a", "b", "c", "d", "e", "f", "g",
        "h", "i", "j", "k", "l", "m", "n", "o", "p", "q", "r", "s",
        "t", "u", "w", "x", "y", "z" };
    ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
        android.R.layout.simple_list_item_multiple_choice, values);
    setListAdapter(adapter);
    getListView().setSelectionMode(ListView.CHOICE_MODE_MULTIPLE);
}
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.main, menu);
    return true;
}
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    Toast.makeText(this,
        String.valueOf(getListView().getCheckedItemCount()),
        Toast.LENGTH_LONG).show();
    return true;
}
}

package com.vogella.android.listview.selection.single;
import android.app.ListActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;
public class MainActivity extends ListActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        String[] values = new String[] { "a", "b", "c", "d", "e", "f", "g",
            "h", "i", "j", "k", "l", "m", "n", "o", "p", "q", "r", "s",
            "t", "u", "w", "x", "y", "z" };
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
            android.R.layout.simple_list_item_single_choice, values);
        setListAdapter(adapter);
        getListView().setSelectionMode(ListView.CHOICE_MODE_SINGLE);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {

```

```

        Toast.makeText(this,
            String.valueOf(getListView().getCheckedItemCount()),
            Toast.LENGTH_LONG).show();
    return true;
}
}

```

## 23- استفاده از ListView برای Contextual action mode (متوجه موقتی که بر روی actionbar اصلی قرار می گیرد)

جهت مطالعه و درک مطالب این بخش، بایستی از قبل با مفهوم ActionBar و contextual action mode آشنا باشید. در زیر به شرح نحوه استفاده از contextual action mode مدیریت حالت انتخاب در ListView خواهیم پرداخت.

به منظور تخصیص یک contextual action mode به یک آیتم از لیست که با کلیک طولانی مدت روی آیتم مورد نظر بر روی اصلی action bar این نمایه (جهت انجام کارهای جزئی تر) نمایان می شود، کافی است متد setOnItemLongClickListener() را بر روی ListView بکار ببرید. این متد اطلاعات آیتم انتخابی را دربر دارد.

مثال زیر استفاده از contextual action mode را به صورت کاربردی به نمایش می گذارد. برای این مثال لازم است یک فایل منو XML به نام rowselection.xml تعریف کرده باشید که منوی آن تنها یک آیتم با شناسه `@+id/menuitem1` show داشته باشد.

```

package de.vogella.android.listactivity;
import android.app.ListActivity;
import android.os.Bundle;
import android.view.ActionMode;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemLongClickListener;
import android.widget.Toast;
public class MyListActivityActionBar extends ListActivity
    implements ActionMode.Callback {
    protected Object mActionMode;
    public int selectedItem = -1;
    @Override
    public void onCreate(Bundle icicle) {

```

```

super.onCreate(icicle);
String[] values = new String[] { "Android",
    "iPhone", "WindowsMobile",
    "Blackberry", "WebOS", "Ubuntu",
    "Windows7", "Max OS X", "Linux", "OS/2",
    "Ubuntu", "Windows7", "Max OS X",
    "Linux", "OS/2", "Ubuntu",
    "Windows7", "Max OS X",
    "Linux", "OS/2" };
MySimpleArrayAdapter adapter = new MySimpleArrayAdapter(this, values);
setListAdapter(adapter);
getListView().setOnItemLongClickListener(new OnItemLongClickListener() {
    @Override
    public boolean onItemLongClick(AdapterView<?> parent, View view, int position, long id) {
        if (mActionMode != null) {
            return false;
        }
        selectedItem = position;
        // Start the CAB using the ActionMode.Callback defined above
        mActionMode =
MyListActivityActionBar.this.startActionMode(MyListActivityActionBar.this);
        view.setSelected(true);
        return true;
    }
});
private void show() {
    Toast.makeText(MyListActivityActionBar.this, String.valueOf(selectedItem),
    Toast.LENGTH_LONG).show();
}
// Called when the action mode is created; startActionMode() was called
@Override
public boolean onCreateActionMode(ActionMode mode, Menu menu) {
    // Inflate a menu resource providing context menu items
    MenuInflater inflater = mode.getMenuInflater();
    // Assumes that you have "contextual.xml" menu resources
    inflater.inflate(R.menu.rowselection, menu);
    return true;
}
// Called each time the action mode is shown. Always called after
// onCreateActionMode, but
// may be called multiple times if the mode is invalidated.
@Override
public boolean onPrepareActionMode(ActionMode mode, Menu menu) {
    return false; // Return false if nothing is done
}
// Called when the user selects a contextual menu item
@Override
public boolean onActionItemClicked(ActionMode mode, MenuItem item) {
    switch (item.getItemId()) {
    case R.id.menuitem1_show:

```

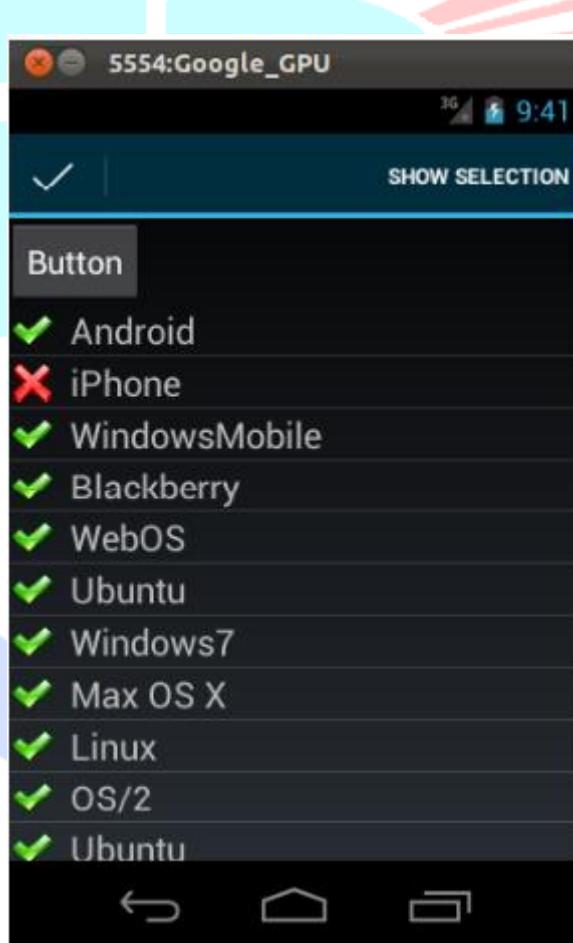
```

        show();
        // Action picked, so close the CAB
        mode.finish();
        return true;
    default:
        return false;
    }
}

// Called when the user exits the action mode
@Override
public void onDismiss(ActionMode mode) {
    mActionMode = null;
    selectedItem = -1;
}
}

```

حال اگر اپلیکیشن خود را اجرا کرده و بر روی یک آیتم در آن طولانی مدت کلیک کنید، **contextual action bar** شما به نمایش در می آید.



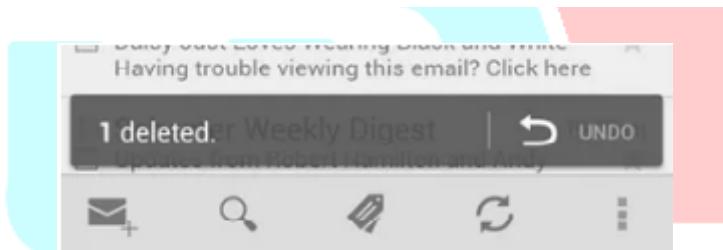
## 24-پیاده سازی قابلیت لغو/undo action

## 2-24-1-چه زمانی باید قابلیت لغو عملیات را به برنامه اضافه کنید

لازم است قابلیت لغو عملیات حساس را برای کاربر اپلیکیشن خود فراهم کنید. برای مثال ممکن است کاربر از حذف آیتم های انتخابی لیست منصرف شده و بخواهد این عملیات را لغو کند.

روش معمول و پر طرفدار این است که یک گزینه در انتهای صفحه برای کاربر به نمایش بگذارد که امکان انتخاب این گزینه با گذشت مدت زمان خاص یا هنگامی که کاربر به ادامه ی تعامل با اپلیکیشن می پردازد، از میان برداشته می شود.

به عنوان نمونه می توان به اپلیکیشن Gmail اشاره کرد که دقیقا همین رفتار را برای لغو عملیات (حذف) پیاده سازی می کند.



**مثال**

نمونه ی زیر قابلیت undo جهت لغو عملیات حذف دائمی یک آیتم را پیاده سازی می کند. برای این منظور یک نوار موقتی نمایان می شود که کاربر برای کلیک بر روی آن مدت زمان محدودی دارد و پس از گذشت آن بازه ی زمانی، نوار حاوی دکمه ی undo با اینیمیشن از صفحه محو می شود.

یک پروژه ی جدید به نام com.vogella.android.userinterface.undo بر اساس قالب آماده BlankTemplate (template) ایجاد کنید.

فایل layout زیر را برای activity خود ایجاد کنید. این فایل با استفاده از FrameLayout دو بخش مجازی رابط کاربری را نمایش می دهد. نواری که دکمه را به نمایش می گذارد، در ابتدا مخفی می باشد. دکمه برای عکس خود از یک drawable استفاده می کند.

```
<FrameLayout 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" >
    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".MainActivity" >
        <ListView
            android:id="@+id/listview"
            android:layout_width="match_parent"
            android:layout_height="match_parent" >
        </ListView>
    </RelativeLayout>
    <LinearLayout
        android:id="@+id/undobar"
        android:visibility="gone"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|center_horizontal"
        android:layout_margin="20dp"
        android:alpha="100"
        android:background="#808080"
        android:dividerPadding="11dp"
        android:padding="4dp" >
        <TextView
            android:id="@+id/undobar_message"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Deleted"    android:textAppearance="?android:attr/textAppearanceMedium"
            android:textColor="#fff" />
        <Button
            android:id="@+id/undobar_button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="30dp"
            android:onClick="onClick"
            android:background="#808080"
            android:drawableLeft="@drawable/ic_undobar_undo"
            android:text="Undo"    android:textAppearance="?android:attr/textAppearanceMedium"
            android:textColor="#fff" />
    </LinearLayout>
</FrameLayout>
```

کلاس activity را به صورت زیر ویرایش کنید. ویزارد پروژه خود یک المان ActionBar تولید می کند. این المان در کد زیر بکار گرفته شده است. در صورت تمایل می توانید ActionBar دلخواه خود را ایجاد کنید.

```
package com.vogella.android.userinterface.undo;
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;
import com.vogella.android.actionbar.undo.R;
public class MainActivity extends Activity {
    private View viewContainer;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ListView l = (ListView) findViewById(R.id.listview);
        String[] values = new String[] { "Ubuntu", "Android", "iPhone", "Windows", "Ubuntu", "Android",
        "iPhone", "Windows" };
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
        android.R.layout.simple_list_item_1, values);
        viewContainer = findViewById(R.id.undobar);
        l.setAdapter(adapter);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.activity_main, menu);
        return true;
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        showUndo(viewContainer);
        return true;
    }
    public void onClick(View view) {
        Toast.makeText(this, "Deletion undone", Toast.LENGTH_LONG).show();
        viewContainer.setVisibility(View.GONE);
    }
    public static void showUndo(final View viewContainer) {
        viewContainer.setVisibility(View.VISIBLE);
        viewContainer.setAlpha(1);
        viewContainer.animate().alpha(0.4f).setDuration(5000)
            .withEndAction(new Runnable() {
```

```
    @Override  
    public void run() {viewContainer.setVisibility(View.GONE);  
    }  
};  
}
```

با انتخاب المان موجود در action bar، نواری حامل دکمه‌ی Undo به مدت ۵ ثانیه نمایش داده می‌شود.



## 25-بهینه سازی کارایی و سرعت اپلیکیشن

نمونه‌ی زیر نسخه‌ی بهینه از adapter مثال قبلی که سرعت و کارایی نسبتاً بهتری دارد را پیاده سازی می‌کند.

ابتدا کلاس زیر به نام MyPerformanceArrayAdapter را ایجاد کنید.

```
package de.vogella.android.listactivity;
import android.app.Activity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
import android.widget.TextView;
public class MyPerformanceArrayAdapter extends ArrayAdapter<String> {
    private final Activity context;
    private final String[] names;
    static class ViewHolder {
        public TextView text;
        public ImageView image;
    }
    public MyPerformanceArrayAdapter(Activity context, String[] names) {
        super(context, R.layout.rowlayout, names);
        this.context = context;
        this.names = names;
    }
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        View rowView = convertView;
        // reuse views
        if (rowView == null) {
            LayoutInflater inflater = context.getLayoutInflater();
            rowView = inflater.inflate(R.layout.rowlayout, null);
            // configure view holder
            ViewHolder viewHolder = new ViewHolder();
            viewHolder.text = (TextView) rowView.findViewById(R.id.TextView01);
            viewHolder.image = (ImageView) rowView
                .findViewById(R.id.ImageView01);
            rowView.setTag(viewHolder);
        }
        // fill data
        ViewHolder holder = (ViewHolder) rowView.getTag();
        String s = names[position];
        holder.text.setText(s);
        if (s.startsWith("Windows7") || s.startsWith("iPhone")
            || s.startsWith("Solaris")) {holder.image.setImageResource(R.drawable.no);
        } else {holder.image.setImageResource(R.drawable.ok);}
    }
}
```

```
    }  
    return rowView;  
}
```

Adapter جدید خود را به activity اضافه کنید. با اجرای برنامه متوجه می شوید که ظاهر آن تغییری نکرده، با این وجود اپلیکیشن به ویژه در خواندن و کار با dataset های بزرگ بسیار سریع تر عمل می کند.

```
package de.vogella.android.listactivity;
import android.app.ListActivity;
import android.os.Bundle;
public class MyListActivity extends ListActivity {
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        String[] values = new String[] { "Android", "iPhone", "WindowsMobile",
            "Blackberry", "WebOS", "Ubuntu", "Windows7", "Max OS X",
            "Linux", "OS/2" };
        setListAdapter(new MyPerformanceArrayAdapter(this, values));
    }
}
```

## 26-آموزش نحوه نمایش دو آیتم در یک ListView

می توانید کلاس SimpleAdapter را جهت نمایش داده های دو المان بکار ببرید. این کلاس به آرایه ای از رشته ها (داده های متغیر آرایه ای from) احتیاج دارد که در آن فیلد هایی که قرار است داده ها از بیرون وارد آن شوند (input data) تعریف شده است. کلاس مزبور همچنین به آرایه ای از اعداد صحیح (int array) نیاز دارد که این اعداد ID یا شناسه ی منحصر بفرد widget هایی در فایل layout می باشند که فیلد ها در سطر به آن نگاشت (map) می شوند.

داده ها در این برنامه لیستی از نوع Map ها (خانواده ای برای نگهداری جفت های کلید و مقدار) است. Map به ازای هر فیلد در From یک مقدار تعریف می کند.

در زیر مثالی را می بینید که برای تنظیم ظاهر سطر مربوطه از یک layout آماده و از پیش ساخته شده هی اندروید استفاده می کند.

```
package de.vogella.android.listactivity;  
import java.util.ArrayList;  
import java.util.HashMap;  
import java.util.Map;  
import android.app.ListActivity;
```

```

import android.os.Bundle;
import android.widget.SimpleAdapter;
public class MyTwoListItemsActivity extends ListActivity {
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        ArrayList<Map<String, String>> list = buildData();
        String[] from = { "name", "purpose" };
        int[] to = { android.R.id.text1, android.R.id.text2 };
        SimpleAdapter adapter = new SimpleAdapter(this, list,
            android.R.layout.simple_list_item_2, from, to);
        setListAdapter(adapter);
    }
    private ArrayList<Map<String, String>> buildData() {
        ArrayList<Map<String, String>> list = new ArrayList<Map<String, String>>();
        list.add(putData("Android", "Mobile"));
        list.add(putData("Windows7", "Windows7"));
        list.add(putData("iPhone", "iPhone"));
        return list;
    }
    private HashMap<String, String> putData(String name, String purpose) {
        HashMap<String, String> item = new HashMap<String, String>();
        item.put("name", name);
        item.put("purpose", purpose);
        return item;
    }
}

```

## 27-2-انتخاب چندین آیتم در ListView

### 27-1-تعامل بین ListView و model

پس از ایجاد لیست، قاعده‌تا می خواهید آیتم‌هایی را در آن انتخاب نمایید. از آنجایی که (به هنگام پیمایش به پایین و بالا لیست) سطرهای ListView بازیافت شده (recycle) و مجدداً مورد استفاده قرار می‌گیرند، نمی‌توان انتخاب (اطلاعات مربوط به وضعیت و انتخاب) را در سطح View ذخیره کرد.



انتخاب آیتم ها در لیست (و تبادل داده بین UI و Model)، تنها یک نمونه از تعامل بین سطح (UI) و داده های متناظر در model است و می توان سناریوهای دیگری را تصور کرد.

به منظور ماندگارسازی و ذخیره ی دائمی انتخاب، شما بایستی data model خود را با اطلاعات مربوط به وضعیت انتخاب شده بروز رسانی نمایید.

به منظور بروز رسانی data model در ListView، ابتدا می بایست کلاس Adapter اختصاصی خود را تعریف نمایید. در سطح این کلاس، شما یک View listener می متصل می کنید که وظیفه ی انتخاب المان متناظر در model را بر عهده دارد. پس از انتخاب المان مورد نظر در لیست، اطلاعات المان مربوطه را در model بروز رسانی می کنید. حال برای بروز رسانی و دسترسی به اطلاعات المان متناظر در model، آن را به صورت یک تگ به view اضافه می کنید.

مثال زیر نشان می دهد چگونه از آبجکت های استاندارد جاوا استفاده کرده و چگونه بین Views و model تعاملو ارتباط دو طرفه برقرار نمایید.

## تمرین آموزشی: تعامل و ارتباط دو طرفه بین Domain model و سطرهای ListView

در این تمرین نیز پروژه ی قبلی، de.vogella.android.listactivity، را ادامه می دهید.  
ابتدا Model زیر را ایجاد نمایید. این کلاس اسم و اطلاعات را، در صورتی که المان در حال حاضر انتخاب شده باشد، در خود ذخیره می نماید.

```
package de.vogella.android.listactivity;
public class Model {
    private String name;
    private boolean selected;
    public Model(String name) {
        this.name = name;
        selected = false;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
```

```

        this.name = name;
    }
    public boolean isSelected() {
        return selected;
    }
    public void setSelected(boolean selected) {
        this.selected = selected;
    }
}
}

```

یک فایل layout به نام rowbuttonlayout.xml زیر ایجاد نمایید.

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >
    <TextView
        android:id="@+id/label"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@+id/label"
        android:textSize="30px" >
    </TextView>
    <CheckBox
        android:id="@+id/check"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_marginLeft="4px"
        android:layout_marginRight="10px" >
    </CheckBox>
</RelativeLayout>

```

یک کلاس Adapter با پیاده سازی زیر ایجاد نمایید. کلاس نام بردہ یک (گوش فراخوان به رخدادها) بر روی آبجکت Checkbox اضافه می کند. به دنبال انتخاب checkbox، داده های مربوطه در model نیز بروز رسانی می شوند. Checkbox به وسیله ی متده getTag()، داده های المان متناظر خود در data model را بازیابی می کند.

```

package de.vogella.android.listactivity;
import java.util.List;
import android.app.Activity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.TextView;
public class InteractiveArrayAdapter extends ArrayAdapter<Model> {
    private final List<Model> list;

```

```

private final Activity context;
public InteractiveArrayAdapter(Activity context, List<Model> list) {
    super(context, R.layout.rowbuttonlayout, list);
    this.context = context;
    this.list = list;
}
static class ViewHolder {
    protected TextView text;
    protected CheckBox checkbox;
}
@Override
public View getView(int position, View convertView, ViewGroup parent) {
    View view = null;
    if (convertView == null) {
        LayoutInflater inflater = context.getLayoutInflater();
        view = inflater.inflate(R.layout.rowbuttonlayout, null);
        final ViewHolder viewHolder = new ViewHolder();
        viewHolder.text = (TextView) view.findViewById(R.id.label);
        viewHolder.checkbox = (CheckBox) view.findViewById(R.id.check);
        viewHolder.checkbox
            .setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
                @Override
                public void onCheckedChanged(CompoundButton buttonView,
                    boolean isChecked) {
                    Model element = (Model) viewHolder.checkbox
                        .getTag();
                    element.setSelected(buttonView.isChecked());
                }
            });
        view.setTag(viewHolder);
        viewHolder.checkbox.setTag(list.get(position));
    } else {
        view = convertView;
        ((ViewHolder) view.getTag()).checkbox.setTag(list.get(position));
    }
    ViewHolder holder = (ViewHolder) view.getTag();
    holder.text.setText(list.get(position).getName());
    holder.checkbox.setChecked(list.get(position).isSelected());
    return view;
}
}

```

حال کد activity را به صورت زیر ویرایش نمایید.

```

package de.vogella.android.listactivity;
import java.util.ArrayList;
import java.util.List;
import android.app.ListActivity;
import android.os.Bundle;
import android.widget.ArrayAdapter;

```

```

public class MyList extends ListActivity {
    /** Called when the activity is first created. */
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        // create an array of Strings, that will be put to our ListActivity
        ArrayAdapter<Model> adapter = new InteractiveListAdapter(this,
            getModel());
        setListAdapter(adapter);
    }
    private List<Model> getModel() {
        List<Model> list = new ArrayList<Model>();
        list.add(get("Linux"));
        list.add(get("Windows7"));
        list.add(get("Suse"));
        list.add(get("Eclipse"));
        list.add(get("Ubuntu"));
        list.add(get("Solaris"));
        list.add(get("Android"));
        list.add(get("iPhone"));
        // Initially select one of the items
        list.get(1).setSelected(true);
        return list;
    }
    private Model get(String s) {
        return new Model(s);
    }
}

```

برنامه را اجرا کرده و آیتم ها را علامت گذاری کنید. با توجه به کد اپلیکیشن، تمامی تغییرات در UI اپلیکیشن، بلافاصله در model منعکس شده و در واقع آبجکت model به صورت خودکار با داده های جدید بروز رسانی می شود.

## 28-پیاده سازی یک ListView کشویی و قابل بسط

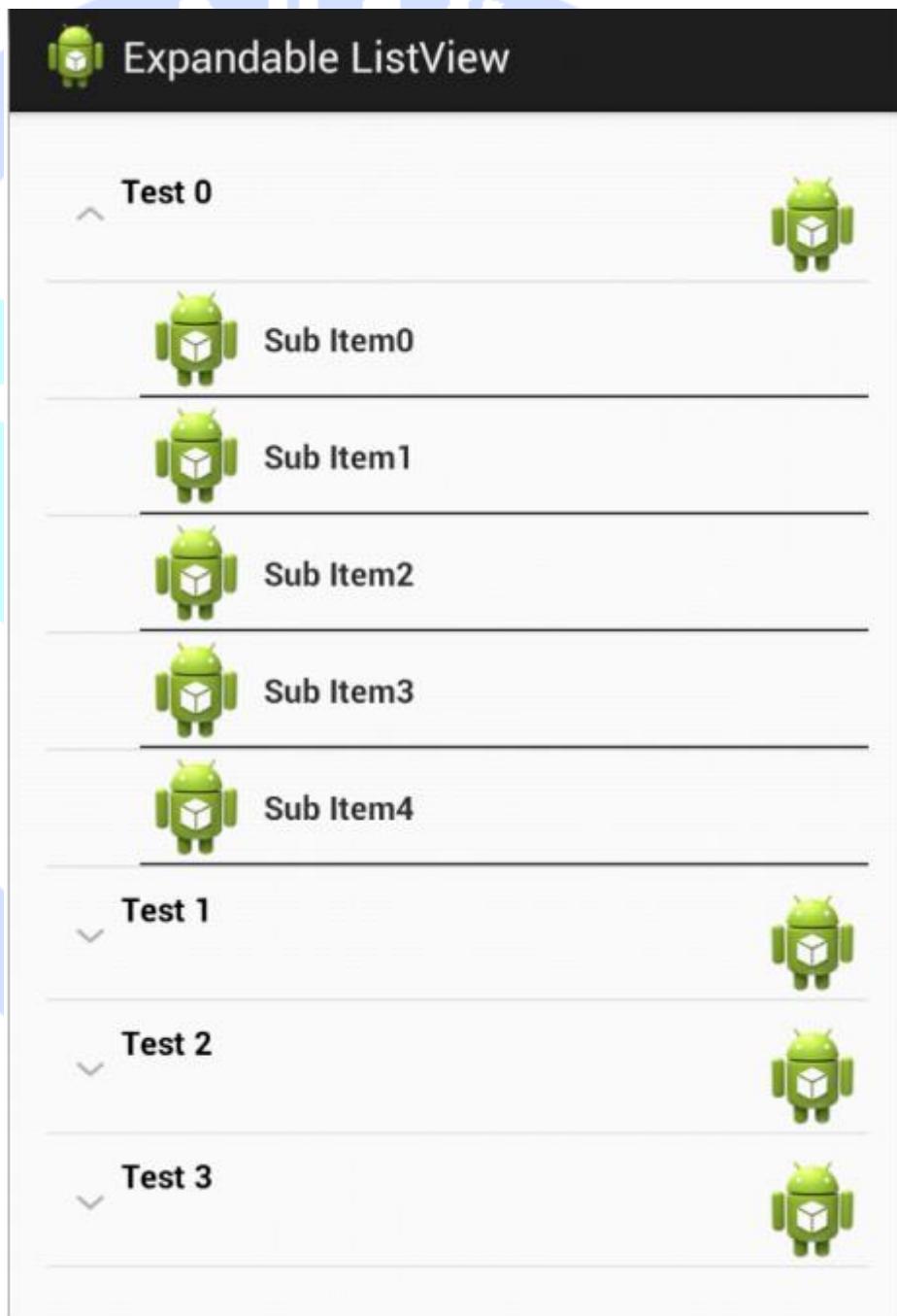
### ExpandableListView-2-28-1

مشابه ListView است با این تفاوت که به شما این امکان را می دهد تا برای المان ها گروه و جزئیات بیشتر ایجاد کنید. برای این منظور group.ExpandableListView از جنس adapter BaseExpandableListAdapter را دارد.

در این سناریو بایستی دو layout تعریف کنید: یکی برای گروه و دیگری برای سطری که جزئیات را نمایش می‌دهد.

### مثالی از پیاده سازی یک ExpandableListView

در این مثال، یک لیست کشویی یا ExpandableListView مشابه تصویر زیر ایجاد خواهد شد.



## 2-28-2-پیاده سازی یک ExpandableListView

یک پروژه جدید به نام com.vogella.android.listview.expandable ای به نام فایل های layout را ایجاد نمایید. ابتدا فایل layout/activity\_main.xml را تعریف کنید.

فایل های layout زیر را ایجاد نمایید. ابتدا فایل layout/activity\_main.xml را تعریف کنید.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >
    <ExpandableListView
        android:id="@+id/listView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >
    </ExpandableListView>
</LinearLayout>
```

سپس فایل layout/listrow\_group.xml را ایجاد نمایید.

```
<CheckedTextView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/textView1"
    android:layout_width="wrap_content"
    android:layout_height="?android:attr/listPreferredItemHeight"
    android:layout_marginLeft="8dp"
    android:drawableRight="@drawable/ic_launcher"
    android:gravity="left"
    android:paddingLeft="32dp"
    android:paddingTop="8dp"
    android:text="Test"
    android:textSize="14sp"
    android:textAlignment="textEnd"
    android:textStyle="bold" />
```

آخرین فایل layout ای که باید تعریف کنید، layout/listrow\_details.xml می باشد.

```
<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="40dp"
```

```

    android:clickable="true"
    android:orientation="vertical"
    android:paddingLeft="40dp"
    tools:context=".MainActivity" >
    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:drawableLeft="@drawable/ic_launcher"
        android:drawablePadding="5dp"
        android:gravity="center_vertical"
        android:text="@string/hello_world"
        android:textSize="14sp"
        android:textStyle="bold" >
    </TextView>
    <View
        android:layout_width="match_parent"
        android:layout_height="1dp"
        android:background="@android:color/black" />
</LinearLayout>
```

کلاس با پیاده سازی زیر را تعریف نمایید که به عنوان domain model آبجکت ExpandableListView شما ایفای نقش می کند.

```

package com.vogella.android.listview.expandable;
import java.util.ArrayList;
import java.util.List;
public class Group {
    public String string;
    public final List<String> children = new ArrayList<String>();
    public Group(String string) {
        this.string = string;
    }
}
```

حال کلاس adapter با پیاده سازی زیر را ایجاد نموده و کد کلاس activity را به صورت زیر ویرایش نمایید.

```

package com.vogella.android.listview.expandable;
import android.app.Activity;
import android.util.SparseArray;
import android.view.LayoutInflater;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup;
import android.widget.BaseExpandableListAdapter;
import android.widget.CheckedTextView;
import android.widget.TextView;
import android.widget.Toast;
```

```

public class MyExpandableListAdapter extends BaseExpandableListAdapter {
    private final SparseArray<Group> groups;
    public LayoutInflater inflater;
    public Activity activity;
    public MyExpandableListAdapter(Activity act, SparseArray<Group> groups) {
        activity = act;
        this.groups = groups;
        inflater = act.getLayoutInflater();
    }
    @Override
    public Object getChild(int groupPosition, int childPosition) {
        return groups.get(groupPosition).children.get(childPosition);
    }
    @Override
    public long getChildId(int groupPosition, int childPosition) {
        return 0;
    }
    @Override
    public View getChildView(int groupPosition, final int childPosition,
                           boolean isLastChild, View convertView, ViewGroup parent) {
        final String children = (String) getChild(groupPosition, childPosition);
        TextView text = null;
        if (convertView == null) {
            convertView = inflater.inflate(R.layout.listrow_details, null);
        }
        text = (TextView) convertView.findViewById(R.id.textView1);
        text.setText(children);
        convertView.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                Toast.makeText(activity, children,
                    Toast.LENGTH_SHORT).show();
            }
        });
        return convertView;
    }
    @Override
    public int getChildrenCount(int groupPosition) {
        return groups.get(groupPosition).children.size();
    }
    @Override
    public Object getGroup(int groupPosition) {
        return groups.get(groupPosition);
    }
    @Override
    public int getGroupCount() {
        return groups.size();
    }
    @Override
    public void onGroupCollapsed(int groupPosition) {
        super.onGroupCollapsed(groupPosition);
    }
}

```

```

@Override
public void onGroupExpanded(int groupPosition) {
    super.onGroupExpanded(groupPosition);
}
@Override
public long getGroupId(int groupPosition) {
    return 0;
}
@Override
public View getGroupView(int groupPosition, boolean isExpanded,
    View convertView, ViewGroup parent) {
    if (convertView == null) {
        convertView = inflater.inflate(R.layout.listrow_group, null);
    }
    Group group = (Group) getGroup(groupPosition);
    ((CheckedTextView) convertView).setText(group.string);
    ((CheckedTextView) convertView).setChecked(isExpanded);
    return convertView;
}
@Override
public boolean hasStableIds() {
    return false;
}
@Override
public boolean isChildSelectable(int groupPosition, int childPosition) {
    return false;
}
}

package com.vogella.android.listview.expandable;
import java.util.ArrayList;
import android.app.Activity;
import android.os.Bundle;
import android.util.SparseArray;
import android.view.Menu;
import android.widget.ExpandableListView;
public class MainActivity extends Activity {
    // more efficient than HashMap for mapping integers to objects
    SparseArray<Group> groups = new SparseArray<Group>();
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        createData();
        ExpandableListView listView = (ExpandableListView) findViewById(R.id.listView);
        MyExpandableListAdapter adapter = new MyExpandableListAdapter(this,
            groups);
        listView.setAdapter(adapter);
    }
    public void createData() {
        for (int j = 0; j < 5; j++) {
            Group group = new Group("Test " + j);
            for (int i = 0; i < 5; i++) {

```

```
        group.children.add("Sub Item" + i);
    }
    groups.append(j, group);
}
}
```

آموزش متفرقه-29-2

۲-۲۹-تعریف کردن یک گوش فراخوان به کلیک طولانی مدت (LongItemClickListener) به آیتم های لیست

بد نیست یک LongItemClickListener به وسیله‌ی setOnItemLongClickListener() برای آیتم‌های لیست خود تعریف کنید که با کلیک طولانی کاربر بر روی آیتم‌های لیست صدا خورده می‌شود.

```
package de.vogella.android.listactivity;
import android.widget.AdapterView.OnItemLongClickListener;
public class MyList extends Activity {
    /** Called when the activity is first created. */
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        // configure your list view as before
        // ListView is assigned to local variable list...
        list.setOnItemLongClickListener(new OnItemLongClickListener() {
            @Override
            public boolean onItemLongClick(AdapterView<?> parent, View view,
                int position, long id) {
                Toast.makeText(MyList.this,
                    "Item in position " + position + " clicked",
                    Toast.LENGTH_LONG).show();
                // Return true to consume the click event. In this case the
                // onListItemClick listener is not called anymore.
                return true;
            }
        });
    }
}
```

## 2-29-2- اضافه کردن Header و Footer به لیست

یا قالب با این چیدمان تعریف نمایید: دو text view تعریف کرده و یک لیست بین این دو قرار

دهید. در این نوع چیدمان یک text view در بالای لیست و در بخش header صفحه و دیگری در پایین text view و در بخش footer قابل مشاهده می باشد. برای نمایش header یا footer در ابتدا و انتهای لیست، کافی است دو متد addFooterView() یا addHeaderView() را در سطح کلاس ListView (بر روی آبجکت listView در قطعه کد زیر) تعریف نمایید.

```
// configuration as before
ListView listView = (ListView) findViewById(R.id.list);
View header = getLayoutInflater().inflate(R.layout.header, null);
View footer = getLayoutInflater().inflate(R.layout.footer, null);
listView.addHeaderView(header);
listView.addFooterView(footer);
listView.setAdapter(new ArrayAdapter<String>(this,
    android.R.layout.simple_list_item_single_choice,
    android.R.id.text1, names));
}
```

## SimpleCursorAdapter-30-2

چنانچه لازم است داده های برنامه ی خود را از Content provider دریافت کنید یا آن را مستقیم از دیتابیس بخوانید، در آن صورت می توانید با استفاده از کلاس SimpleCursorAdapter اطلاعات مورد نیاز لیست (ListView) خود را مشخص نمایید (داده ها را از view به cursor متصل نمایید). توضیحات زیر برای شما شرح می دهد چگونه می توان به content provider متعلق نمایید. یک دیتابیس مقایس پذیر با اطلاعات مربوط به مخاطبین یا contacts کند) دسترسی داشت و اطلاعات مورد نیاز لیست را در اختیار آن قرار داد ( یادآوری: provider به شما اجازه می دهد تا داده های خود را در نقطه ی مرکزی برای دسترسی اپلیکیشن هایی که به آن نیاز دارند به اشتراک بگذارید).

یک پروژه ی جدید اندروید به نام "de.vogella.android.listactivity.cursor" به همراه activity ای به نام MyListActivity تعریف نمایید. کلاس MyListActivity را به صورت زیر ویرایش نمایید.

```
package de.vogella.android.listactivity.cursor;
import android.app.ListActivity;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.provider.ContactsContract;
```

```

import android.widget.ListAdapter;
import android.widget.SimpleCursorAdapter;
public class MyListActivity extends ListActivity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Cursor mCursor = getContacts();
        startManagingCursor(mCursor);
        // now create a new list adapter bound to the cursor.
        // SimpleListAdapter is designed for binding to a Cursor.
        ListAdapter adapter = new SimpleCursorAdapter(this, // Context,
            android.R.layout.two_line_list_item, // Specify the row template
            mCursor, // Pass in the cursor to bind to.
            // Array of cursor columns to bind to.
            new String[] { ContactsContract.Contacts._ID,
                ContactsContract.Contacts.DISPLAY_NAME },
            // Parallel array of which template objects to bind to those
            // columns.
            new int[] { android.R.id.text1, android.R.id.text2 });
        // Bind to our new adapter.
        setListAdapter(adapter);
    }
    private Cursor getContacts() {
        // Run query
        Uri uri = ContactsContract.Contacts.CONTENT_URI;
        String[] projection = new String[] { ContactsContract.Contacts._ID,
            ContactsContract.Contacts.DISPLAY_NAME };
        String selection = ContactsContract.Contacts.IN_VISIBLE_GROUP + " = ''"
            + ("1") + "";
        String[] selectionArgs = null;
        String sortOrder = ContactsContract.Contacts.DISPLAY_NAME
            + " COLLATE LOCALIZED ASC";
        return managedQuery(uri, projection, selection, selectionArgs,
            sortOrder);
    }
}

```

لازم است به اپلیکیشن خود مجوز خواندن اطلاعات contacts را بدهید (مقدار خصیصه ی روی بر manifest یا در فایل تنظیمات را android:name: قرار دهید).

## 31-سایر کتابخانه های کد باز (Open Source libraries)

کاربر مجبور است برای بروز رسانی ActionBar بر روی دکمه `refresh` کلیک کند که ممکن است گاهی تجربه ی کاربری ضعیفی را به دنبال داشته باشد. Chris Banes یک کتابخانه ی کد باز پیاده سازی کرده که الگوی `pull to refresh` را برای ListView در اپلیکیشن شما implement می کند. برای دسترسی به آن می توانید به این آدرس مراجعه نمایید: <https://github.com/chrisbanes/Android-PullToRefresh>

علاوه بر آن شاید لازم باشد با پیاده سازی قابلیت `swipe` در اپلیکیشن، به کاربر این امکان را بدهید که آیتم ها را با کشیدن انگشت بر روی نمایشگر از لیست حذف کند. Roman Nurik برای این منظور یک کتابخانه ی آماده نوشته و Jake Wharton این کتابخانه را برای پشتیبانی از ورژن های قبلی کتابخانه های اندروید (و افزودن قابلیت های جدید به API های قدیمی تر) backport نموده است. برای دسترسی به هریک می توانید به ترتیب به آدرس های <https://github.com/romannurik/android-swipetodismiss> و <https://github.com/JakeWharton/SwipeToDismissNOA> مراجعه نمایید.

بخش هفتم :

## 32- ثبت وقایع و گزارش گیری (Logging) در اندروید

## ثبت وقایع و گزارش گیری با استفاده از ابزار Android logging / LogCat

آموزش حاضر نحوه‌ی ایجاد و استفاده از دستورات log در اپلیکیشن‌های اندروید را شرح می‌دهد.

### 2-33-1- سیستم گزارش گیری اندروید (log system)

اندروید برای ثبت گزارشات (جهت اشکال زدایی پروژه) از یک سیستم مرکزی بهره می‌گیرد. البته برنامه نویس این امکان را نیز دارد که پیغام‌های دلخواه خود را تنظیم کند. ابزاری که برای توسعه و طراحی اندروید در اختیار شما قرار می‌گیرند به شما این امکان را می‌دهند تا با تعریف فیلتر تنها از دستوراتی که کارایی مد نظر شما را ارائه می‌دهد، استفاده نمایید.

### 2-33-2- ایجاد دستورات گزارش گیری (log statement)

به منظور استفاده از دستورات گزارش گیری (log statement)، می‌بایست از کلاس android.util.Log و متدهای زیر استفاده نمایید.

- Log.v()
- Log.d()
- Log.i()
- Log.w()
- Log.e()
- Log.wtf()

این توابع بر اساس اولویت و میزان کاربرد فهرست شده اند به طوری که () Log.i از کم ترین میزان اهمیت و کاربرد برخوردار است. اولین پارامتر ورودی این متدها category و دومین آرگومان پیغام مربوطه می‌باشد.

معمولاً برنامه نویس ابتدا یک الگو پیاده سازی (interface) Constants در اپلیکیشن اندرویدی خود تعریف می‌کند و سپس log category را در قالب یک فیلد در بدنه‌ی آن interface ارائه می‌دهد.

```
// package declaration left out, use your application package
public interface Constants {
    String LOG = "com.vogella.testapp";
}
```

اندروید به توسعه دهندگان توصیه می کند که از داشتن کدهای مربوط به گزارشات و وقایع (logging) در اپلیکیشن نصب شده (deployed) خودداری کنند. مجموعه ابزار طراحی و توسعه ADT برای این منظور BuildConfig.DEBUG (flag) را ارائه می دهد. زمانی که اپلیکیشن اندرویدی خود را به صورت یک خروجی امضا شده (export)، نهایی و آماده برای نصب منتشر می کنید، مقدار این flag به صورت خودکار بر روی false تنظیم می شود. در طول توسعه ای اپلیکیشن، مقدار این flag بر روی true قرار می گیرد، بدین معنا که برنامه نویس می تواند دستورات و کدهای مربوط به گزارش از برنامه را حین نوشتن کدهای برنامه مشاهده کند.

مثال زیر نشان می دهد چگونه یک پیغام خطأ را بنویسید.

```
if (BuildConfig.DEBUG) {
    Log.e(Constants.TAG, "onCreate called");
}
```

: بخش هشتم

## 34-پیاده سازی dialog در اپلیکیشن به وسیلهٔ DialogFragments

## نمایش پنجره‌ی محاوره dialog به وسیله‌ی DialogFragments

این آموزش نحوه‌ی پیاده‌سازی و استفاده از dialog یا پنجره‌ی محاوره‌ای (که به صورت کادر شناور بر روی پنجره‌ی activity نمایش داده می‌شود) در اپلیکیشن‌های اندرویدی را شرح می‌دهد.

پروژه‌ی این مبحث داخل محیط کاری Android Studio نوشته شده و مبتنی بر ویرایش 5.0 سیستم عامل اندروید می‌باشد.

### 34-2-به نمایش گذاشتن dialog در برنامه

جهت به نمایش گذاشتن یک کادر محاوره‌ای در Activity اپلیکیشن، بایستی نمونه‌ای از کلاس DialogFragment را پیاده‌سازی نمایید. در واقع یک fragment (آبجکتی که تکه ای از یک activity کل را تشکیل می‌دهد) است که به برنامه نویس این امکان را می‌دهد تا پنجره‌ی محاوره‌ای را به صورت شناور بر روی پنجره‌ی اصلی که activity اپلیکیشن مربوطه می‌باشد، به نمایش بگذارد.

به منظور به نمایش گذاشتن dialog، می‌توانید یا یک کلاس dialog از پیش‌پیاده‌سازی شده در fragment را باز گردانی نمایید و یا یک layout اختصاصی برای آن تعریف کنید که به هنگام اجرا برنامه فراخوانی و به نمایش گذاشته می‌شود.

### 34-2-استفاده از dialog های آماده و از قبل موجود

برنامه شما می‌تواند متد onCreateDialog را پیاده‌سازی نموده و dialog از DialogFragment بازگردانی کند. کلاس Dialog، در واقع کلاس پایه یا پدر است که برای پیاده‌سازی dialog باید از آن ارث بری نمایید.

برنامه نویس برای نمایش دادن dialog در اپلیکیشن خود اغلب یکی از کلاس های مشتق شده از آن همچون TimePickerDialog، AlertDialog، ProgressDialog یا DatepickerDialog را مورد استفاده قرار می دهد.

اندروید همچنین یک ProgressDialog ارائه می دهد که به راحتی با فراخوانی متد `open()` می توان آن را باز کرده و به نمایش گذاشت.

### 2-34-3- تعریف DialogFragment برای layout اختصاصی

جهت ایجاد dialog اختصاصی خود، لازم است یک فایل layout ویژه‌ی آن تعریف نمایید. این فایل layout را می توان inflate نموده و با فراخوانی متد `onCreateView()` از fragment مربوطه بازگردانی نمود.

### 2-34-4- تعامل activity با DialogFragment

آبجکت DialogFragment درست مشابه دیگر fragment ها با activity مربوطه ارتباط برقرار کرده و اطلاعاتی را رد و بدل می کند. یک روش صحیح و بهینه این است که یک fragment معرفی کرده و مورد نظر این interface activity را پیاده سازی می کند. سپس می تواند توسط activity صدا خورده شود، بدون اینکه اطلاعی از جزئیات پیاده سازی activity داشته باشد.

## تمرین: پیاده سازی DialogFragment کلاس

در پروژه‌ی زیر نحوه‌ی استفاده از کلاس DialogFragment در activity را شرح خواهیم داد. این مثال هم از dialog از پیش ساخته و آماده استفاده می کند و هم برای آن یک layout اختصاصی تعریف می نماید.

## 34-2-ایجاد پروژه و فایل های layout

ابتدا یک پروژه اندروید به نام com.android.vogella.com.dialogfragmentexample ایجاد نمایید.

در فایل layout ای که فراخوانی می کند، دو دکمه تعریف کنید به طوری که در خصیصه onClick خود به متاد android:onClick (attribute) اشاره کنند.

حال یک فایل layout با بدنی زیر تعریف نمایید.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/edit_name"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:orientation="vertical">
    <TextView
        android:id="@+id/lbl_your_name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Enter user name" />
    <EditText
        android:id="@+id/username"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:imeOptions="actionDone"
        android:inputType="text" />
</LinearLayout>
```

## 34-2-تعریف fragment و تنظیم activity

دو کلاس fragment زیر را ایجاد نمایید.

```
package dialogfragmentexample.android.vogella.com.dialogfragmentexample;
import android.app.AlertDialog;
import android.app.Dialog;
import android.app.DialogFragment;
import android.content.DialogInterface;
import android.os.Bundle;
import android.widget.Toast;
public class MyAlertDialogFragment extends DialogFragment {
    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        return new AlertDialog.Builder(getActivity())
            // set dialog icon
            .setIcon(android.R.drawable.stat_notify_error)
            // set Dialog Title
            .setTitle("Alert dialog fragment example")
```

```

    // Set Dialog Message
    .setMessage("This is a message")
    // positive button
    .setPositiveButton("OK", new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int which) {
            Toast.makeText(getActivity(), "Pressed OK", Toast.LENGTH_SHORT).show();
        }
    })
    // negative button
    .setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int which) {
            Toast.makeText(getActivity(), "Cancel", Toast.LENGTH_SHORT).show();
        }
    }).create();
}
}

```

```

package dialogfragmentexample.android.vogella.com.dialogfragmentexample;
import android.app.DialogFragment;
import android.os.Bundle;
import android.view.KeyEvent;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.WindowManager;
import android.widget.EditText;
import android.widget.TextView;
public class MyDialogFragment extends DialogFragment implements TextView.OnEditorActionListener {
    private EditText mEditText;
    public interface UserNameListener {
        void onFinishUserDialog(String user);
    }
    // Empty constructor required for DialogFragment
    public MyDialogFragment() {
    }
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                           Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_username, container);
        mEditText = (EditText) view.findViewById(R.id.username);
        // set this instance as callback for editor action
        mEditText.setOnEditorActionListener(this);
        mEditText.requestFocus();
        getWindow().setSoftInputMode(
                WindowManager.LayoutParams.SOFT_INPUT_STATE_VISIBLE);
        setTitle("Please enter username");
        return view;
    }
    @Override
    public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {
        // Return input text to activity

```

```

        UserNameListener activity = (UserNameListener) getActivity();
        activity.onFinishUserDialog(mEditText.getText().toString());
        this.dismiss();
        return true;
    }
}

```

کد activity خود را به صورت زیر ویرایش نمایید.

```

package dialogfragmentexample.android.vogella.com.dialogfragmentexample;
import android.app.Activity;
import android.app.Fragment;
import android.app.FragmentManager;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;
public class MainActivity extends Activity implements MyDialogFragment.UserNameListener {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    @Override
    public void onFinishUserDialog(String user) {
        Toast.makeText(this, "Hello, " + user, Toast.LENGTH_SHORT).show();
    }
    public void onClick(View view) {
        // close existing dialog fragments
        FragmentManager manager = getFragmentManager();
        Fragment frag = manager.findFragmentByTag("fragment_edit_name");
        if (frag != null) {
            manager.beginTransaction().remove(frag).commit();
        }
        switch (view.getId()) {
            case R.id.showCustomFragment:
                MyDialogFragment editNameDialog = new MyDialogFragment();
                editNameDialog.show(manager, "fragment_edit_name");
                break;
            case R.id.showAlertDialogFragment:
                MyAlertDialogFragment alertDialogFragment = new MyAlertDialogFragment();
                alertDialogFragment.show(manager, "fragment_edit_name");
                break;
        }
    }
}

```

## تست اپلیکیشن

اکنون زمانی که اپلیکیشن را اجرا نموده و بر روی دکمه‌ی مربوطه در UI اپلیکیشن کلیک می‌کنید، پنجره‌ی محاوره‌ای (dialog) حاوی پیغام مورد نظر برای شما به نمایش در می‌آید.





## بخش اول :

### طراحی صفحات تک قطعه و چند قطعه (& single-pane) با استفاده از fragment ها در اندروید/ایجاد الاهای انعطاف پذیر به وسیله ای fragment ها

این مبحث برای شما شرح می دهد چگونه می توان با استفاده از کلاس fragment، اپلیکیشن هایی با رابط کاربری مقیاس و انعطاف پذیر ایجاد نمایید.

#### Fragment-1-3

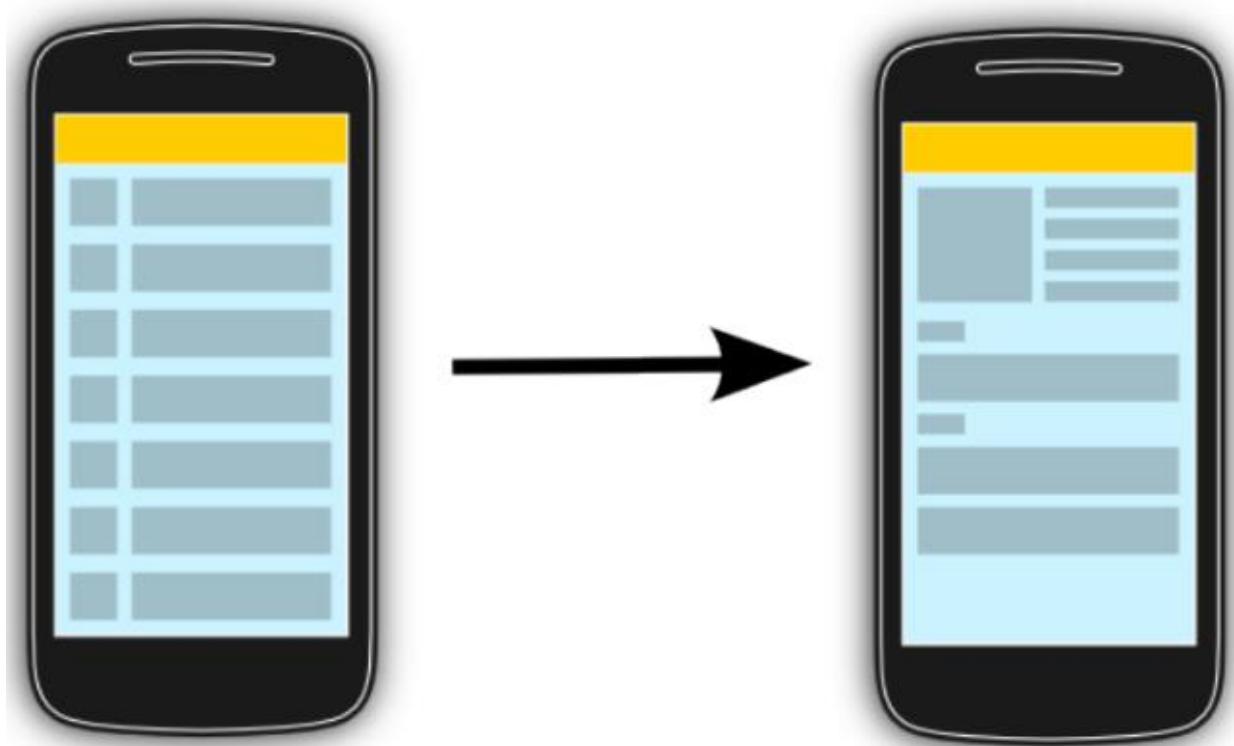
##### 3-1-1-شرح Layout های تک قطعه (single-pane) و چند قطعه (multi-pane)

همان طور که می دانید دستگاه های اندروید دارای نمایشگرهایی با اندازه و تراکم پیکسلی متفاوت هستند.

Panel یا pane در اندروید، عبارت است از یک بخش یا قطعه از کل صفحه (UI) که کاربر با آن تعامل دارد. pane در واقع یک واژه‌ی کلی است که بیانگر قابلیت اندروید برای پشتیبانی از چندین view در کنار هم و در قالب یک view مرکب (واحد) بوده که ممکن است بسته به اندازه‌ی فضای موجود در نمایشگر از دستگاه به دستگاهی دیگر متفاوت باشد.

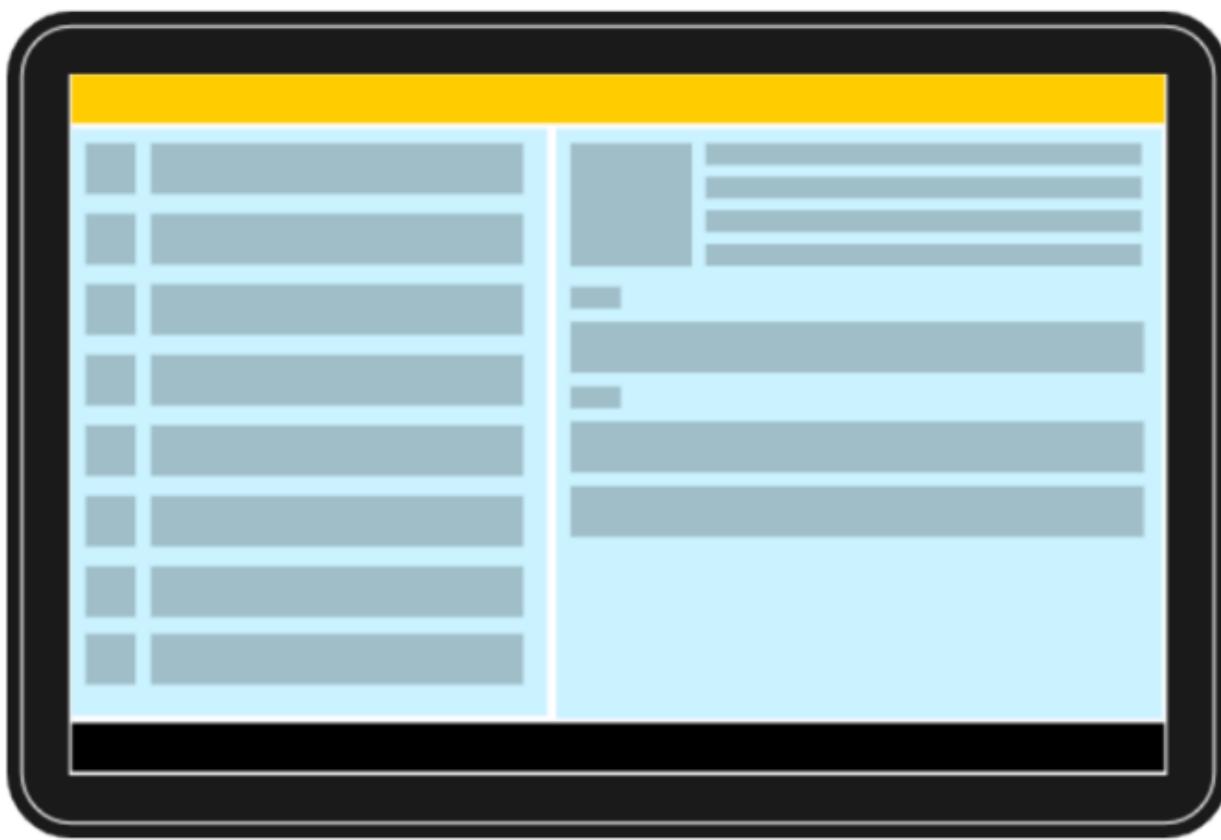


در صورت عدم وجود فضای کافی، تنها یک panel به نمایش در می آید. از این معمولاً تحت عنوان چیدمان یا قالب (layout) تک قطعه نام برده می شود.



اما در دستگاه هایی که صفحه‌ی نمایش عریض‌تری دارند و در نتیجه فضای بیشتری در دسترس است (همچون تبلت)، طراحی چند قطعه این امکان را برای برنامه فراهم می‌کند تا چندین panel را همزمان به نمایش بگذارد.

## آموزشگاه تحلیلکردن و داده



### 3-1-2- شرح مفهوم Fragment

activity در حقیقت یک کامپوننت UI مستقل در اندروید است که زیرمجموعه‌ی یک fragment می‌باشد و بخشی از کل صفحه یا فرم را تشکیل می‌دهد (در نمایشگرهای بزرگ چند fragment و در نمایشگرهای کوچک تنها یک fragment در لحظه نمایش داده شده و صفحه‌ی قابل مشاهده برای کاربر را تشییک می‌دهند). به عبارت دیگر fragment در اندروید به شما اجازه می‌دهد تا UI انعطاف‌پذیر و سازگار با انواع نمایشگرهای کوچک و بزرگ (کوچک و بزرگ) برای برنامه‌ی خود تعریف نمایید و اجزا رابط کاری اپلیکیشن را در قالب مازول‌های مجزا کپسوله سازی نمایید.

Fragment با کپسوله سازی اجزا رابط کاربری و رفتار activity‌ها در مازول‌های مجزا، این امکان را برای اپلیکیشن فراهم می‌کند تا از این مازول‌های مجزا در چندین activity استفاده کند.

در بستر یک activity اجرا می شود، با این وجود دارای چرخه‌ی حیات و رابط کاربری اختصاصی خود است. البته یک fragment می تواند قادر ظاهر و UI باشد که در اصطلاح خوانده headless fragment می شود.

### Context ها و دسترسی به Fragment-3-1-3

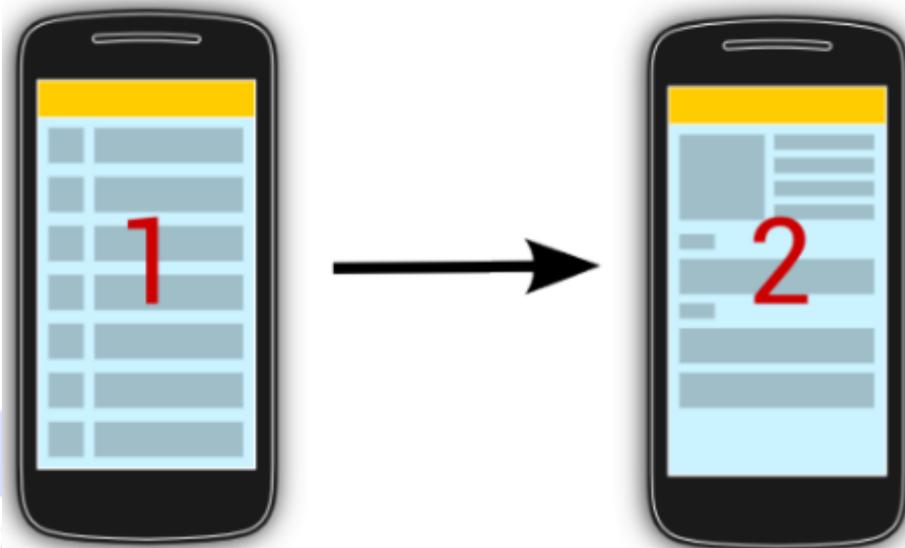
activity ها از کلاس Context مشتق (subclass) نمی شوند. بنابراین برای دسترسی به activity میزبان بایستی متند getActivity() را صدا بزنید.

### fragment-3-1-4 مزایای استفاده از

1. fragment ها چرخه‌ی حیات و رفتار خود را دارند.
2. می توان آن ها را در زمان اجرای اپلیکیشن و به صورت داینامیک کم و زیاد کرد.
3. می توان چندین fragment را با یکدیگر ترکیب کرده و UI های چند قطعه/چند پنجره‌ای و انعطاف پذیر ایجاد کرد.
4. این امکان نیز وجود دارد که یک fragment را در چندین activity بکار برد.

به کمک fragment می توانید برای دستگاه های کوچک UI یک قطعه ایجاد نموده و برای نمایشگرهای بزرگ رابط کاربری چند پنجره طراحی نمایید. همچنین می توانید با استفاده از fragment هر دو جهت نمایش (portrait =نمای افقی و landscape =عمودی) در گوشی ها را مدیریت نمایید.

یکی از موارد کاربرد fragment در طراحی UI را می توان در ساخت لیست مشاهده کرد. اگر بر روی یک آیتم از لیست کلیک کنید، در صفحه‌ی تبلت، جزئیات مربوطه در همان صفحه، برای مثال در سمت راست به نمایش در می آیند. اما همین لیست در نمایشگر موبایل، کاربر را به صفحه‌ی مجزا حاوی جزئیات مرتبط هدایت می کند.



برای این مبحث اپلیکیشن شما باید دو fragment داشته باشد: main و detail. البته شما می توانید بسته به نیاز خود fragment های بیشتری تعریف نمایید. بعلاوه اپلیکیشن شما می بایست دو activity به شرح زیر داشته باشد: main activity و detailed activity. در نمایشگر تبلت هر دو fragment را در layout خود به نمایش می گذارد. این در حالی است که در صفحه نمایش موبایل، فقط main fragment را در لحظه به نمایش می گذارد.

آموزشگاه تخلیکر داده ها



280

آدرس آموزشگاه: تهران - خیابان شریعتی - بالاتر از خیابان ملک - جنب بانک صادرات - پلاک 651 طبقه دوم - واحد 7

88146323 - 88446780 - 88146330

### ۵-۳-۱-۵- طراحی اپلیکیشنی با UI انعطاف پذیر و سازگار با نمایشگرهای مختلف به وسیلهٔ fragment ها

Fragment ها را می‌توان به صورت static تعریف کنید، بدین معنی که به وسیلهٔ تگ `<fragment>` در فایل layout می‌توانید مشخص کنید که از چندین activity و قطعهٔ fragment تشکیل شود.

همان‌طور که گفته شد، شما می‌توانید fragment های یک activity را در زمان اجرا ویرایش کنید (کم یا زیاد کنید) که به آن تعریف به صورت پویا و dynamic definition گفته می‌شود.

برای اینکه بتوانید با توجه به فضای موجود در نمایشگر خود چند fragment را نمایش دهید، کافی است بر اساس یکی از روش‌های زیر اقدام نمایید:

۱. از یک activity استفاده کنید که در نمایشگرهای بزرگ (تبلت) و کوچک (گوشی همراه) دو fragment را برای کاربر در لحظه نمایش می‌دهد و در زمان لازم، مشاهده در activity (صفحهٔ جاری) را به هنگام اجرای برنامه تغییر دهید. برای این منظور توصیه می‌شود دو نمونه از کلاس FrameLayout به عنوان placeholder (مکان نگهدار موقتی) در layout خود تعریف نموده، سپس fragment ها را در زمان اجرا به آن دو کلاس اضافه کنید.

- در دستگاه های کوچک همچون گوشی، هر fragment را در activity جدا بگنجانید. برای مثال، اگر UI تبلت از دو fragment در یک activity استفاده می‌کند و در آن واحد دو قطعه را به نمایش می‌گذارد، همین activity را برای نمایشگر گوشی استفاده می‌کنید اما این بار فایل layout مجزا ارائه می‌دهید که تنها یک fragment را شامل می‌شود. چنانچه main activity fragment از قبل در نمایشگر حاضر باشد، آنگاه fragment به main activity دستور بروز رسانی خود را می‌دهد. اگر detail fragment در دسترس نبود، در آن صورت main activity را راه اندازی می‌کند.

اینکه کدام گزینه را انتخاب کنید کاملاً به نیاز و مورد استفادهٔ شما بستگی دارد. به طور معمول ویرایش fragment در زمان اجرا منعطف‌تر است، اما پیاده سازی آن کمی پیچیده‌تر می‌باشد.

## 2-3-تعریف و استفاده از fragment ها

### 2-3-1-تعریف fragment

به منظور تعریف یک fragment جدید می‌توانید از کلاس android.app.Fragment و یا یکی از کلاس‌های فرزند (subclass) آن ارث بری (extend) نمایید. کلاس‌های فرزندی که می‌توانید مورد استفاده قرار دهید عبارتند از: PreferenceFragment، DialogFragment، ListFragment یا WebViewFragment.

کد زیر یک نمونه پیاده سازی را نمایش می‌دهد:

```
package com.example.android.rssreader;
import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
public class DetailFragment extends Fragment {
    public static final String EXTRA_URL = "url";
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                           Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_rssitem_detail,
                           container, false);
        return view;
    }
    @Override
    public void onActivityCreated(Bundle savedInstanceState) {
        super.onActivityCreated(savedInstanceState);
        Bundle bundle = getArguments();
        if (bundle != null) {
            String link = bundle.getString("url");
            setText(link);
        }
    }
    public void setText(String url) {
        TextView view = (TextView) getView().findViewById(R.id.detailsText);
        view.setText(url);
    }
}
```

## ۲-۳-۲-۲-تعامل اپلیکیشن با fragment ها

برای اینکه بتوان امکان استفاده ای مجدد از fragment ها را افزایش داد و به اصطلاح آن ها را بازیافت نمود، لازم است هر گونه تعامل بین این fragment ها منحصر از طریق activity میزبان انجام شود. به بیان دیگر، fragment ها نباید مستقیماً و بدون واسطه با یکدیگر ارتباط داشته باشند.

برای این منظور fragment می بایست یک interface را به صورت inner و داخل یک کلاس تعریف کند. ایجاد fragment می کند که activity میزبان، این interface را پیاده سازی کند. بدین وسیله شما اطمینان حاصل می کنید که هیچ اطلاعی از activity میزبان (که از آن onAttach() استفاده می کند) خود ندارد. Fragment مورد نظر سپس به وسیله ای متده استفاده می کند که آیا activity مورد نظر را پیاده سازی می کند یا خیر.

به عنوان مثال، فرض کنید ای دارد که باید مقداری را به activity میزبان (parent) خود ارسال کند. این کار را می توان به صورت زیر پیاده سازی نمود.

```
package com.example.android.rssreader;
import android.app.Activity;
import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
public class MyListFragment extends Fragment {
    private OnItemSelectedListener listener;
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_rsslist_overview,
            container, false);
        Button button = (Button) view.findViewById(R.id.button1);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                updateDetail("fake");
            }
        });
        return view;
    }
}
```

```

public interface OnItemSelectedListener {
    public void onRssItemSelected(String link);
}

@Override
public void onAttach(Context context) {
    super.onAttach(context);
    if (context instanceof OnItemSelectedListener) {
        listener = (OnItemSelectedListener) context;
    } else {
        throw new ClassCastException(context.toString()
            + " must implement MyListFragment.OnItemSelectedListener");
    }
}

@Override
public void onDetach() {
    super.onDetach();
    listener = null;
}

// may also be triggered from the Activity
public void updateDetail(String uri) {
    // create a string just for testing
    String newTime = String.valueOf(System.currentTimeMillis());
    // inform the Activity about the change based
    // interface definition
    listener.onRssItemSelected(newTime);
}
}

```

### ۳-۲-۳- ارسال پارامتر به fragment ها

Activity می تواند یک bundle را به عنوان پارامتر به fragment ارسال کند.

```

detailFragment = new DetailFragment();
// configure link
Bundle bundle = new Bundle();
bundle.putString("link", link);
detailFragment.setArguments(bundle);

```

آبجکت fragment را در متده `onActivityCreated` خود دریافت می کند.

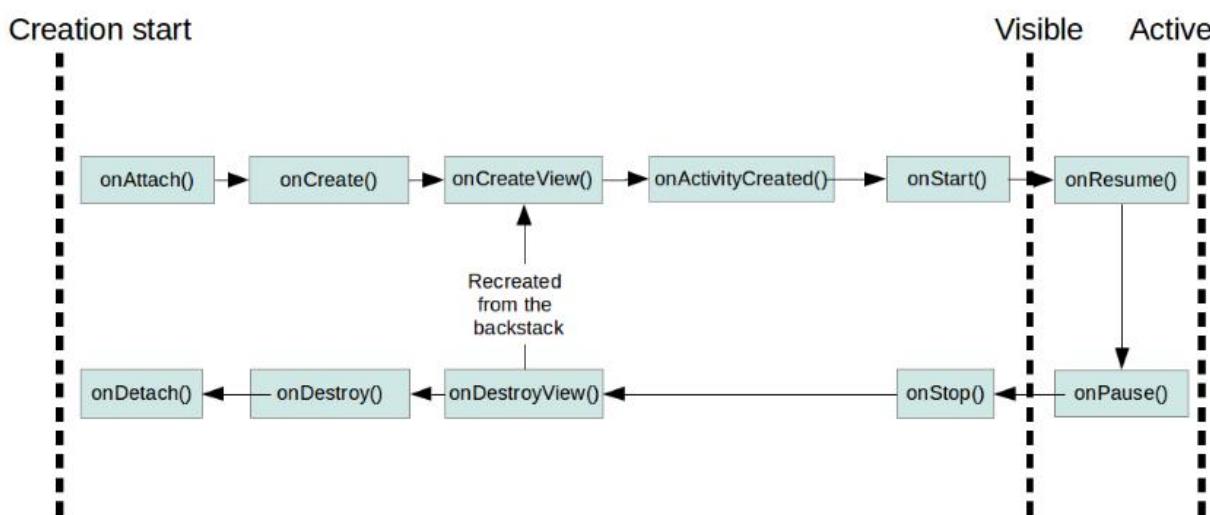
```

@Override
public void onActivityCreated(Bundle savedInstanceState) {
    super.onActivityCreated(savedInstanceState);
    Bundle bundle = getArguments();
    if (bundle != null) {
        setText(bundle.getString("link"));
    }
}

```

### 3-3-چرخه‌ی حیات (life cycle) fragment

در اندروید هر fragment هر چرخه‌ی حیات مختص به خود را دارد. با این وجود چرخه‌ی حیات آن همیشه به activity میزبان آن متصل است.



زمانی که activity متوقف می‌شود، همگام با آن های fragment زیرمجموعه‌ی این activity نیز متوقف می‌شوند. اگر activity از حافظه کلاپک شود (destroy)، های fragment آن نیز همزمان از بین می‌روند.

متدها	شرح کارکرد
onAttach()	کاربرد متدهای این است که fragment را به activity می‌شناساند. زمانی که onAttach() اجرا شود، fragment و activity صدا خورده‌اند. شما می‌توانید داخل هیچ یک ایجاد نشده‌اند. این متدهای این که می‌خواهید پیش از ایجاد fragment انجام شود، تعریف نمایید. نمونه‌ی activity به نمونه‌ی fragment متصل است. به هنگام فراخوانی این متدهای activity و fragment، هنوز راه اندازی نشده‌اند. معمولاً برنامه نویس اشاره‌گری به activity داخل این

متد	شرح کارکرد
	متد قرار می دهد که از fragment برای مقداردهی بیشتر استفاده می کند.
onCreate()	<p>اندروید زمانی این متد را صدا می زند که در حال ساختن fragment باشد. برنامه نویس می تواند کامپوننت هایی از fragment را که می خواهد در زمان ادامه یافتن activity، بعد از توقف حفظ شود را مقداردهی اولیه نمایید. در واقع در این مرحله، fragment ایجاد شده است. این متد بعد از فراخوانده شدن() مربوط به activity و قبل از متد onCreateView() مربوط به fragment خورده می شود.</p>
onCreateView()	<p>نمونه‌ی fragment، زنجیره یا سلسله مراتب view های خود را ایجاد می کند. به عبارت دیگر onCreateView() در متد fragment کاربری یا ظاهر خود را تعریف می کند. در این متد با فراخوانی متد inflate() از آجکت Inflater که به عنوان پارامتر به این متد فرستاده می شود، برنامه نویس می تواند یک layout را بارگذاری کرده و نمایش (inflate) دهد.</p> <p>در این متد نباید با activity تعامل داشته باشید. Activity هنوز کاملا ساخته و مقداردهی اولیه نشده است.</p> <p>لزومی ندارد این متد را برای fragment های headless فراخوانی کنید. فاقد UI های view شده بخشی از زنجیره ی view های activity میزبان محسوب می شوند. به عبارت دیگر، هنگامی که fragment می خواهد view های خود را بسازد این متد را صدا</p>

متد	شرح کارکرد
	<p>می زند. به منظور پیاده سازی UI یا رابط کاری، برنامه نویس می بایست یک آبجکت view که پوشه‌ی حاوی fragment را مشخص می کند، به عنوان خروجی این متد تعریف کنید. اگر رابط کاربری نداشته باشد، می توانید خروجی متد را null تعریف کنید.</p>
<code>onActivityCreated()</code>	<p>متد جاری پس از فراخوانی متد <code>onCreateView()</code> و زمانی که activity میزبان ایجاد شده است، صدا خورده می شود.</p>
	<p>نمونه‌ی fragment و Activity همراه با زنجیره‌ی view های activity ساخته شده اند. در این برهه، می توان با فراخوانی متد <code>findViewById()</code> دلخواه دسترسی داشت.</p> <p>در این متد شما می توانید از آبجکت هایی که به یک آبجکت Context نیاز دارند، نمونه سازی نمایید.</p> <p>به عبارت دیگر، این متد بعد از <code>onCreateView()</code> و هنگامی که activity میزبان ایجاد شده، <code>call</code> می شود. و activity به صورت یک آبجکت view سلسه مراتبی برای <code>activity</code> تولید می شوند.</p>
<code>onStart()</code>	<p>The <code>onStart()</code> method is called once the fragment gets visible.</p> <p>متد <code>onStart()</code> زمانی صدا زده می شود که fragment نمایان و قابل مشاهده شود.</p>
<code>onResume()</code>	<p>زمانی که Fragment در حال اجرا است.</p>

متد	شرح کارکرد
<code>onPause()</code>	<p>قابل مشاهده است اما دیگر فعال نیست مانند زمانی که activity دیگری، با fragment activity حامل اینیمیشن بر روی قرار می گیرد.</p> <p>با کاربر تعاملی ندارد. حال یا آن در حال متوقف شدن است یا یک عملیات مربوط به fragment در حال ویرایش آن در activity است.</p>
<code>onStop()</code>	<p>دیگر برای کاربر قابل مشاهده نیست.</p>
<code>onDestroyView()</code>	<p>یا المان های رابط کاربری قابل مشاهد fragment را پاک می کند. اگر در ساخته شده باشد، در آن صورت از backstack این متد فراخوانی می شود و پس از آن onCreateView صدا زده می شود. در واقع پس از فراخوانی این متد UI و ظاهر به کلی از بین می رود.</p>
<code>onDestroy()</code>	<p>این متد cleanup و پاک سازی نهایی fragment ها را انجام می دهد. لازم به ذکر است که این متد ممکن است اصلا فراخوانی نشود.</p>

### activity برای fragment تعریف 4-3

### 3-4-3- تعریف fragment برای layout خود در فایل activity به صورت static

برای استفاده از fragment جدید خود، می توانید آن را به صورت static و به وسیله ی تگ fragment در فایل layout اضافه نمایید. خصیصه ی (attribute) android:name به کلاس fragment مربوطه اشاره دارد.



```
<?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="fill_parent"
    android:layout_height="fill_parent"
    android:baselineAligned="false"
    android:orientation="horizontal" >
    <fragment
        android:id="@+id/listFragment"
        android:layout_width="0dp"
        android:layout_weight="1"
        android:layout_height="match_parent"
        class="com.example.android.rssreader.MyListFragment"
        tools:layout="@layout/fragment_rsslist_overview">
    </fragment>
    <fragment
        android:id="@+id/detailFragment"
        android:layout_width="0dp"
        android:layout_weight="2"
        android:layout_height="match_parent"
        class="com.example.android.rssreader.DetailFragment"
        tools:layout="@layout/fragment_rssitem_detail">
    </fragment>
</LinearLayout>
```

استفاده از این روش برای زمانی توصیه می شود که چندین فایل layout ثابت و static برای تعریف ظاهر برنامه دارید که هریک ویژه ی یک دستگاه با config خاص طراحی شده است.

### 3-4-2- مدیریت fragment در زمان اجرا و به صورت dynamic

کلاس FragmentManager به شما این امکان را می دهد تا fragment ها را در activity layout اضافه/حذف یا جایگزین نمایید. این کلاس به وسیله ی متد getFragmentManager() قابل دسترسی می باشد. اصلاحات و تغییراتی که می خواهید اعمال کنید باید در طی یک transaction (تراکنش) به وسیله ی کلاس FragmentTransaction انجام شود. به منظور ویرایش

ها در یک activity می بایست یک مکان نگهدار (placeholder) در فایل layout در فایل activity تعریف نمایید.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal" >
    <FrameLayout
        android:id="@+id/listcontainer"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
    <FrameLayout
        android:id="@+id/detailscontainer"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:visibility="gone" />
</LinearLayout>
```

با استفاده از fragmentManager می توانید fragment موجود در container را با دیگری جایگزین نمایید.

```
// get fragment manager
FragmentManager fm = getSupportFragmentManager();
// add
FragmentTransaction ft = fm.beginTransaction();
ft.add(R.id.your_placeholder, new YourFragment());
// alternatively add it with a tag
// trx.add(R.id.your_placeholder, new YourFragment(), "detail");
ft.commit();
// replace
FragmentTransaction ft = fm.beginTransaction();
ft.replace(R.id.your_placeholder, new YourFragment());
ft.commit();
// remove
Fragment fragment = fm.findFragmentById(R.id.your_placeholder);
FragmentTransaction ft = fm.beginTransaction();
ft.remove(fragment);
ft.commit();
```

یک جدید جایگزین fragment جاری در این container می شود.

می توانید یک تراکنش یا transaction را با استفاده از متده addBackStack() به backstack اضافه نمایید. با این کار عملیات را به history stack از activity مورد نظر اضافه می کنید و به کاربر این امکان را می دهد تا تغییرات را از طریق دکمه back (بازگشت) به حالت قبلی بازگرداند.

### 4-3-بررسی اینکه آیا یک fragment در layout حاضر است یا خیر

به منظور بررسی اینکه آیا یک fragment در layout مورد نظر حضور دارد یا خیر، می توانید از کلاس FragmentManager کمک بگیرید. کافی است متدهای `isLayout()` را فراخوانی کرده و بررسی کنید آیا fragment مورد نظر از طریق layout به activity اضافه شده است یا خیر.

```
DetailFragment fragment = (DetailFragment) getSupportFragmentManager().  
    findFragmentById(R.id.detail_frag);  
if (fragment==null || !fragment.isInLayout()) {  
    // start new Activity  
}  
else {  
    fragment.update(...);  
}
```

### 4-4-بررسی تعداد fragment ها

منطق برنامه در activity به طور قطع به سناریو و شرایط جاری بستگی دارد (اینکه آیا صفحه تک قطعه است یا از چند fragment و قطعه تشکیل شده). می توان با نوشتن یک قطعه کد شرطی به تعداد fragment در اپلیکیشن پی برد. روش های متعددی برای اجرای این عملیات وجود دارد. روش اول این است که یک فایل تنظیمات/`configuration` در پوششی محتوا و منابع پروژه (`values`) ایجاد نمایید. جفت کلید/مقدار (`key/value`) به صورت پیش فرض بر روی `false` تنظیم می شوند. سپس یک فایل تنظیمات دیگر این مقدار را جهت سازگاری با اندازه ی دلخواه نمایشگر بر روی `true` قرار می دهد.

نمونه ی زیر یک فایل تنظیمات پیش فرض به نام `config.xml` را نمایش می دهد.

```
<resources>  
    <item type="bool" name="twoPaneMode">false</item>  
</resources>
```

حال همین فایل را در پوششی `res/values-land` با مقداری متفاوت (این بار `true`) ایجاد نمایید.

```
<resources>
    <item type="bool" name="twoPaneMode">true</item>
</resources>
```

می توانید با استفاده نوشتن دستور زیر، به اطلاعات مربوط به وضعیت جاری (state) دسترسی داشته باشید.

```
getResources().getBoolean(R.bool.twoPaneMode);
```

### 4-3-5-افزودن آبجکت به fragment transition

می توانید آبجکت backstack را به FragmentTransition را به اضافه نموده و به کاربر این امکان را بدهید تا با استفاده از دکمه‌ی بازگشت (back) به fragment قبلی بازگردد.

برای این منظور شما می توانید متد () addBackStack() را بر روی آبجکت FragmentTransition را فراخوانی کنید.

### 4-3-6-استفاده از افکت‌های تعریف شده توسط API برای حرکت بین fragment‌ها

در طول اجرای یک تراکنش fragment، می توانید با استفاده از متد () setCustomAnimations() از مجموعه توابع Property Animation animation تعریف نمایید (برای حرکت و انتقال بین fragment‌ها انیمیشن اعمال نمایید).

البته می توانید با فراخوانی متد () setTransition()، بسیاری از انیمیشن‌های استاندارد اندروید را برای انتقال بین fragment‌ها اعمال نمایید. برای پیاده‌سازی انیمیشن می بایست constant‌هایی که با \* FragmentTransaction.TRANSIT\_FRAGMENT\_\* شروع می شوند را تعریف نمایید.

هر دو متد نام بردۀ به شما امکان می دهند تا یک انیمیشن آغاز (entry animation) و یک انیمیشن پایان (exit) تعریف نمایید.

### 4-3-ماندگارسازی و ذخیره ی دائمی داده در fragment ها

اغلب برنامه نویس لازم دارد که داده هایی از اپلیکیشن را در حافظه به طور دائمی ذخیره کند. برای این منظور کافی است از یک فایل ساده یا دیتابیس SQLite استفاده کند.

### 4-3-نگهداری و بازگردانی اطلاعات پس از تغییر در تنظیمات و config

اگر می خواهید تغییرات در تنظیمات و پیکربندی را ماندگارسازی نمایید، در آن صورت می توانید از آجکت اپلیکیشن نیز استفاده کنید.

علاوه بر روش ذکر شده، شما می توانید متده استفاده از `setRetainState(true)` را بر روی fragment ماندگار می شود. با این کار اطلاعات مربوط به وضعیت نمونه ی fragment بین تغییرات در تنظیمات fragment ماندگار می شود. لازم به ذکر است که این رویکرد تنها زمانی کارگر واقع می شود که fragment ها به backstack اضافه نشده باشند. در این صورت، کافی داده ها را به عنوان یک عضو (در قالب یک فیلد) ذخیره نمایید.

**توجه:** Google استفاده از این متده را برای fragment هایی که UI دارند به هیچ وجه توصیه نمی کند.

می توانید با فراخوانی متده `onSaveInstanceState()` داده ها را در آجکت Bundle قرار دهید. سپس می توانید این داده ها را با استفاده از متده `onActivityCreated()` بازیابی نمایید.

### background ها و پردازش در پس زمینه (processing)

#### headless fragments/UI های فاقد Fragment-3-5-1

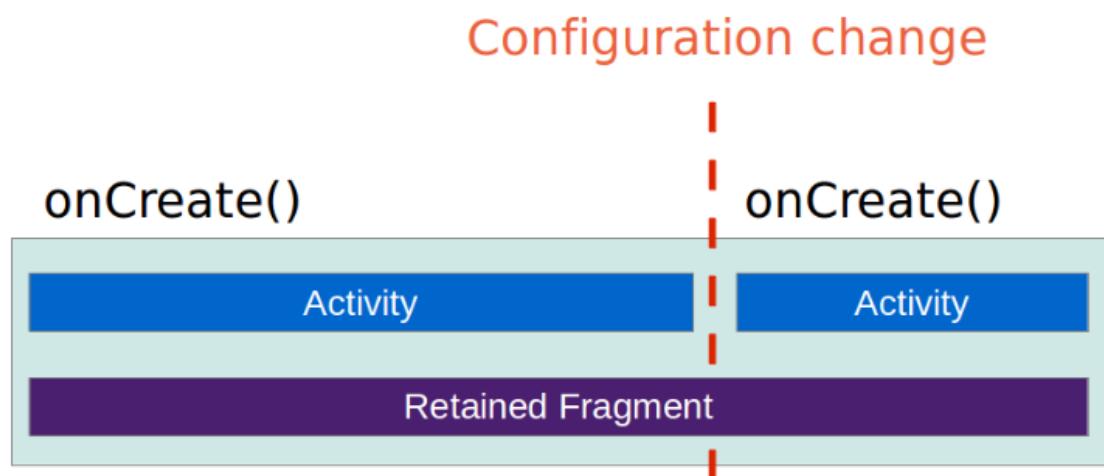
می توانید fragment هایی تعریف نمایید که اصلا UI و ظاهر ندارند. به این fragment ها در اصطلاح headless گفته می شود. جهت پیاده سازی چنین fragment ای کافی است مقدار null را از متده `onCreateView()` در برگردانید.

**توجه:** توصیه می شود برای پردازش پس زمینه ای از service ها استفاده نمایید. اگر می خواهید این کار را از طریق fragment های خود انجام دهید، دومین راه حل پیشنهادی ما (بعد از

سرویس ها) headless fragment همراه با فراخوانی متدهای `setRetainInstance()` هست. بدین وسیله دیگر شما مجبور نیستید تغییرات در تنظیمات را حین پردازش ناهم زمان (asynchronous) (processing)، خود مدیریت نمایید.

### 3-5-2 Retained headless fragment - تنظیمات (config changes)

Headless fragment ها معمولاً به منظور کپسوله سازی (encapsulate) و نگهداری اطلاعات مربوط به وضعیت fragment ها، مابین تغییرات در تنظیمات و نحوه پیکربندی اپلیکیشن، استفاده می شوند. مورد کاربرد دیگری که می توان به آن اشاره کرد: برای task هایی که پردازش هایی را در پس زمینه انجام می دهند (background processing task) نیز مورد استفاده قرار می گیرند. برای کپسوله سازی داده ها در fragment ها، می توانید headless fragment را retained fragment تعریف نمایید. از بین نمی روند (در واقع این fragment ها قادرند اشاره گرها بی به آبجکت های پایدار و stateful که شما می خواهید اطلاعات آن ها را حفظ نمایید، نزد خود نگه دارند).



برای اینکه fragment اطلاعات مربوط به وضعیت را نگه دارند و به اصطلاح retained باشند، کافی است متدهای `setRetainInstance()` را صدا بزنید.

به منظور افزودن چنین activity ای به fragment مورد نظر، می توانید متد () add() از کلاس FragmentManager را فراخوانی کنید. برای اینکه در آینده به این Fragment دسترسی داشته باشید (به آن اشاره کنید)، لازم است آن را با یک تگ در فایل XML اضافه نمایید. این کار به شما اجازه می دهد، با استفاده از متد () findFragmentByTag() از FragmentManager به آن دسترسی داشته باشید.

توجه داشته باشید که استفاده از متد () onRetainNonConfigurationInstance() در activity دیگر منسوخ شده است و جای خود را به retained headless fragment ها داده اند.

## تمرین: پیاده سازی fragment ها در اپلیکیشن ها

مبحث زیر نحوه استفاده از fragment ها در یک اپلیکیشن متعارف اندروید، بدون استفاده از support library (کتابخانه ای پشتیبانی از API های جدید در ویرایش های قدیمی تر اندروید) را تشریح می کند. اپلیکیشن بسته به حالت نمایش (نمای افقی/عمودی)، از layout با fragment های متفاوت بهره می گیرد.

در نمای عمودی، RssfeedActivity در لحظه تنها یک fragment را به نمایش می گذارد. از این fragment، کاربر می تواند به activity دیگری که میزبان fragment مورد نظر است راه پیدا کند. در نمای افقی، activity مزبور هر دو fragment را درکنار هم برای کاربر به نمایش می گذارد.

## ایجاد پروژه

یک پروژه ای جدید اندروید ایجاد نموده و آن را به صورت زیر مقداردهی نمایید. توجه داشته باشید که نباید از لایه ای سازگاری (Compatibility layer) استفاده کنید.

### جدول 1

Property	مقدار
----------	-------

Application Name	RSS Reader
Company Domain (اسم دامنه ی شرکت)	android.example.com
Package name (اسم)	com.example.android.rssreader
Template (الگو یا قالب آماده)	Empty Activity
Activity	RssfeedActivity
Layout	activity_rssfeed

### ۵-۳- تعریف فایل های layout برای fragment ها

یک فایل layout جدید به نام res/layout/fragment\_rssitem\_detail.xml ایجاد نمایید.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <TextView
        android:id="@+id/detailsText"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="center_horizontal|center_vertical"
        android:layout_marginTop="20dip"
        android:text="Default Text"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:textSize="30dip" />
</LinearLayout>
```

**توجه:** id باید صحیح باشد چرا که fragment با استفاده از این شناسه به المان TextView دسترسی دارد.

یک فایل layout جدید به نام res/layout/fragment\_rsslist\_overview.xml ایجاد نمایید.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <Button
        android:id="@+id/updateButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Press to update"
        />
</LinearLayout>
```

**توجه:** در اینجا هم id باید دقیق باشد.

### 4-5-3-ایجاد کلاس های fragment

کلاس های زیر را ایجاد نمایید. این کلاس ها به عنوان fragment مورد استفاده قرار می گیرند.  
ابتدا کلاس DetailFragment را تعریف نمایید.

```
package com.example.android.rssreader;
import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
public class DetailFragment extends Fragment {
    public static final String EXTRA_URL = "url";
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                           Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_rssitem_detail,
                                     container, false);
        return view;
    }
    @Override
    public void onActivityCreated(Bundle savedInstanceState) {
        super.onActivityCreated(savedInstanceState);
        Bundle bundle = getArguments();
        if (bundle != null) {
            String link = bundle.getString("url");
            setText(link);
        }
    }
    public void setText(String url) {
        TextView view = (TextView) getView().findViewById(R.id.detailsText);
        view.setText(url);
    }
}
```

کلاس myListFragment را ایجاد نمایید. این کلاس برخلاف اسمش، هیچ لیستی را به نمایش نمی گذارد، بلکه صرفا یک دکمه ارائه می دهد که با کلیک کاربر بر روی آن، مقدار زمان جاری به ای به نام DetailsFragment ارسال می گردد.

```
package com.example.android.rssreader;
import android.app.Fragment;
import android.content.Context;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
```

```
import android.view.ViewGroup;
import android.widget.Button;
public class MyListFragment extends Fragment {
    private OnItemSelectedListener listener;
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                           Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_rsslist_overview,
                                     container, false);
        Button button = (Button) view.findViewById(R.id.updateButton);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                updateDetail("fake");
            }
        });
        return view;
    }
    public interface OnItemSelectedListener {
        void onRssItemSelected(String link);
    }
    @Override
    public void onAttach(Context context) {
        super.onAttach(context);
        if (context instanceof OnItemSelectedListener) {
            listener = (OnItemSelectedListener) context;
        } else {
            throw new ClassCastException(context.toString()
                + " must implement MyListFragment.OnItemSelectedListener");
        }
    }
    // triggers update of the details fragment
    public void updateDetail(String uri) {
        // create fake data
        String newTime = String.valueOf(System.currentTimeMillis());
        // send data to activity
        listener.onRssItemSelected(newTime);
    }
}
```

### ۳-۵-۵-ویرایش فایل layout اصلی

فایل activity\_rssfeed.xml فعالی را ویرایش نمایید.

```
<?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="fill_parent"
    android:layout_height="fill_parent"
    android:baselineAligned="false"
    android:orientation="horizontal" >
    <fragment
```

```

    android:id="@+id/listFragment"
    android:layout_width="0dp"
    android:layout_weight="1"
    android:layout_height="match_parent"
    class="com.example.android.rssreader.MyListFragment"
    tools:layout="@layout/fragment_rsslist_overview">
</fragment>
<fragment
    android:id="@+id/detailFragment"
    android:layout_width="0dp"
    android:layout_weight="2"
    android:layout_height="match_parent"
    class="com.example.android.rssreader.DetailFragment"
    tools:layout="@layout/fragment_rssitem_detail">
</fragment>
</LinearLayout>

```

### 5-3-6- ویرایش کلاس RssfeedActivity

کلاس RssfeedActivity را ویرایش کنید به طوری که نقش یک تابع callback (بازفرارخوان) را برای ایفا کند و متعاقبا DetailsFragment را بروز رسانی نماید.

```

package com.example.android.rssreader;
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
public class RssfeedActivity extends Activity implements MyListFragment.OnItemSelectedListener{
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_rssfeed);
    }
    @Override
    public void onRssItemSelected(String link) {
        DetailFragment fragment = (DetailFragment) getSupportFragmentManager()
            .findFragmentById(R.id.detailFragment);
        fragment.setText(link);
    }
}

```

### تست اپلیکیشن و کسب اطمینان از اجرای موفقیت آمیز آن

اپلیکیشن خود را اجرا نمایید. هر دو fragment باید در نمای افقی و عمودی نمایش داده شوند. می توانید با استفاده از کنترل های شبیه ساز (emulator)، جهت نمایش را تغییر دهید. به دنبال فشرده شدن دکمه در DetailFragment، اطلاعات ListFragment بروز آوری می شوند.



## تمرین: نمایش fragment ها بر اساس تنظیمات و نحوه پیکربندی

چنانچه اپلیکیشن در نمای افقی راه اندازی شود، قاعدها باید دو fragment (دو قطعه یا pane) را در UI برای کاربر به نمایش بگذارد. در تمرین حاضر اپلیکیشن را طوری تنظیم کنید که از این قابلیت یا رفتار پشتیبانی کند.

300

آدرس آموزشگاه: تهران - خیابان شریعتی - بالاتر از خیابان ملک - جنب بانک صادرات - پلاک 651 طبقه دوم - واحد 7

88146323 - 88446780 - 88146330

### 3-5-7- تعریف activity layout ویژه‌ی نمای عمودی

فایل activity\_rssfeed.xml را با تنظیمات زیر تعریف نمایید.

```
<?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"
    tools:context=".RssfeedActivity"
    android:orientation="vertical">
    <FrameLayout
        android:id="@+id/fragment_container"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:orientation="horizontal"
        android:layout_weight="1"
        android:paddingBottom="@dimen/activity_vertical_margin"
        android:paddingLeft="@dimen/activity_horizontal_margin"
        android:paddingRight="@dimen/activity_horizontal_margin"
        android:paddingTop="@dimen/activity_vertical_margin"
    >
    </FrameLayout>
</LinearLayout>
```

### 3-5-8- تعریف یک flag یا گزینه‌ی بولی مستقل از

(گزینشگر منابع)

یک فایل به نام config.xml در پوشه‌ی res/values با تنظیمات زیر ایجاد نمایید.

```
<resources>
    <item type="bool" name="twoPaneMode">false</item>
</resources>
```

همین فایل را در پوشه‌ی res/values-land با مقداری متفاوت تعریف نمایید.

```
<resources>
    <item type="bool" name="twoPaneMode">true</item>
</resources>
```

### 3-5-9- تنظیم و ویرایش کلاس RssfeedActivity

پیاده سازی کلاس RssfeedActivity را طوری ویرایش نمایید که در صورت قرار گرفتن اپلیکیشن در وضعیت تک قطعه/قابه (single-pane) fragment جاری را جایگزین کند.

```

package com.example.android.rssreader;
import android.app.Activity;
import android.app.Fragment;
import android.app.FragmentManager;
import android.os.Bundle;
import android.widget.FrameLayout;
public class RssfeedActivity extends Activity implements
    MyListFragment.OnItemSelectedListener {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_rssfeed);
        if (getResources().getBoolean(R.bool.twoPaneMode)) {
            // all good, we use the fragments defined in the layout
            return;
        }
        // if savedInstanceState is null we do some cleanup
        if (savedInstanceState != null) {
            // cleanup any existing fragments in case we are in detailed mode
            fragmentManager().executePendingTransactions();
            Fragment fragmentById = fragmentManager().  

                findFragmentById(R.id.fragment_container);
            if (fragmentById!=null) {
                fragmentManager().beginTransaction()
                    .remove(fragmentById).commit();
            }
        }
        MyListFragment listFragment = new MyListFragment();
        FrameLayout viewById = (FrameLayout) findViewById(R.id.fragment_container);
        fragmentManager().beginTransaction()
            .replace(R.id.fragment_container, listFragment).commit();
    }
    @Override
    public void onRssItemSelected(String link) {
        if (getResources().getBoolean(R.bool.twoPaneMode)) {
            DetailFragment fragment = (DetailFragment) fragmentManager()
                .findFragmentById(R.id.detailFragment);
            fragment.setText(link);
        } else {
            // replace the fragment
            // Create fragment and give it an argument for the selected article
            DetailFragment newFragment = new DetailFragment();
            Bundle args = new Bundle();
    }
}

```

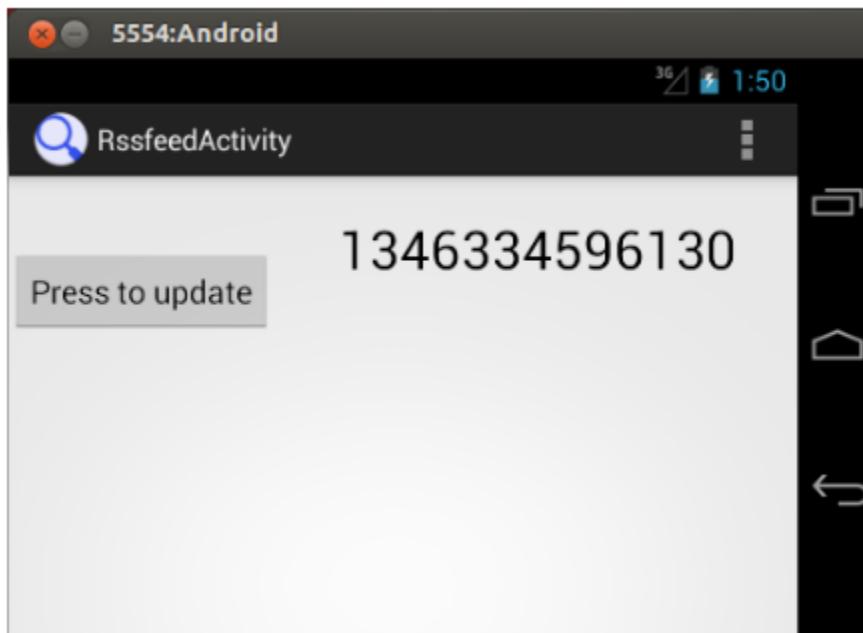
```
args.putString(DetailFragment.EXTRA_URL, link);
newFragment.setArguments(args);
FragmentTransaction transaction = getFragmentManager().beginTransaction();
// Replace whatever is in the fragment_container view with this fragment,
// and add the transaction to the back stack so the user can navigate back
transaction.replace(R.id.fragment_container, newFragment);
transaction.addToBackStack(null);
// Commit the transaction
transaction.commit();
}
}
```

**توجه:** این فرایند پاک سازی ضروری است، چراکه اپلیکیشن ما به صورت dynamic و در زمان اجرا بین دو حالت تک قطعه یا نمایش یک fragment در لحظه و نمایش دو قطعه یا fragment در آن واحد، تغییر وضعیت می دهد. البته سناریوی حاضر غیرمعمول بوده و معمولاً انتخاب یک fragment برای اپلیکیشن بستگی به حداقل عرض نمایشگر دارد و این پیکربندی در زمان اجرای برنامه تغییر نمی یابد.FragmentManager سعی دارد با cache یا ذخیره یی موقتی fragment کارایی را افزایش دهد، از اینرو لازم است detailed fragment موجود را حذف نمایید.

تست اپلیکیشن و کسب اطمینان از درستی پیاده سازی آن

اپلیکیشن را اجرا نمایید. اگر اپلیکیشن را در نمای عمودی (portrait mode) اجرا کنید، طبیعتاً باید تنها یک fragment را در نمایشگر مشاهده نمایید. در نمای افقی (landscape mode) شما هر دو fragment را در همزمان در UI مشاهده خواهید نمود.

حال وضعیت نمایش شبیه ساز را تغییر دهید. بر روی دکمه‌ی مورد نظر در نمای عمودی (portrait) و سپس در نمای افقی (landscape mode) کلیک نمایید. اطمینان حاصل نمایید که activity، زمان جاری را نمایش می‌دهد.



تمرین: ذخیره‌ی اطلاعات مربوط به وضعیت در یک

**retained fragment**

در تمرین حاضر قصد دارید که آخرین انتخاب کاربر را ذخیره کرده و بباد آورید. برای این منظور می‌بایست از یک **headless fragment** استفاده نمایید.

**ایجاد headless fragment**

```
package com.example.android.rssreader;
import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
public class SelectionStateFragment extends Fragment {
    public String lastSelection = "";
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState) {
        return null;
```

```

    }
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setRetainInstance(true);
    }
}
}

```

### 3-5-3- ذخیره ی آخرین مقدار انتخابی در یک headless fragment

```

package com.example.android.rssreader;
import android.app.Activity;
import android.app.Fragment;
import android.app.FragmentManager;
import android.os.Bundle;
import android.widget.FrameLayout;
public class RssfeedActivity extends Activity implements
    MyListFragment.OnItemSelectedListener {
    SelectionStateFragment stateFragment;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_rssfeed);
        stateFragment =
            (SelectionStateFragment) getFragmentManager()
                .findFragmentByTag("headless");
        if(stateFragment == null) {
            stateFragment = new SelectionStateFragment();
            getFragmentManager().beginTransaction()
                .add(stateFragment, "headless").commit();
        }
        if (findViewById(R.id.fragment_container) == null) {
            // restore state
            if (stateFragment.lastSelection.length() > 0) {
                onRssItemSelected(stateFragment.lastSelection);
            }
            // all good, we use the fragments defined in the layout
            return;
        }
        // if savedInstanceState is null we do some cleanup
        if (savedInstanceState != null) {
            // cleanup any existing fragments in case we are in detailed mode
            getFragmentManager().executePendingTransactions();
            Fragment fragmentById = getFragmentManager().
                findFragmentById(R.id.fragment_container);
            if (fragmentById!=null) {
                getFragmentManager().beginTransaction()
                    .remove(fragmentById).commit();
            }
        }
        MyListFragment listFragment = new MyListFragment();
        FrameLayout viewById = (FrameLayout) findViewById(R.id.fragment_container);

```

```

getFragmentManager().beginTransaction()
    .replace(R.id.fragment_container, listFragment).commit();
}
@Override
public void onRssItemSelected(String link) {
    stateFragment.lastSelection = link;
    if (getResources().getBoolean(R.bool.twoPaneMode)) {
        DetailFragment fragment = (DetailFragment) getFragmentManager()
            .findFragmentById(R.id.detailFragment);
        fragment.setText(link);
    } else {
        // replace the fragment
        // Create fragment and give it an argument for the selected article
        DetailFragment newFragment = new DetailFragment();
        Bundle args = new Bundle();
        args.putString(DetailFragment.EXTRA_URL, link);
        newFragment.setArguments(args);
        FragmentTransaction transaction = getFragmentManager().beginTransaction();
        // Replace whatever is in the fragment_container view with this fragment,
        // and add the transaction to the back stack so the user can navigate back
        transaction.replace(R.id.fragment_container, newFragment);
        transaction.addToBackStack(null);
        // Commit the transaction
        transaction.commit();
    }
}
}
}

```

### ۳-۵-۳- تعریف یک flag یا گزینه‌ی بولی مستقل از گزینشگر resource

یک فایل در پوشه‌ی res/values به نام config.xml با تنظیمات زیر تعریف نمایید.

```

<resources>
    <item type="bool" name="dualPane">false</item>
</resources>

```

همین فایل را در پوشه‌ی res/values-land با مقداری متفاوت ایجاد نمایید.

```

<resources>
    <item type="bool" name="dualPane">true</item>
</resources>

```

### ۳-۵-۴- تنظیم و ویرایش کلاس RssfeedActivity

بدنه‌ی کلاس RssfeedActivity را طوری ویرایش نمایید که در صورت قرار گرفتن اپلیکیشن در وضعیت تک قطعه (single-pane)، fragment جاری را جایگزین کند.

```
package com.example.android.rssreader;
```

```

import android.app.Activity;
import android.app.Fragment;
import android.app.FragmentManager;
import android.os.Bundle;
import android.widget.FrameLayout;
public class RssfeedActivity extends Activity implements
    MyListFragment.OnItemSelectedListener {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_rssfeed);
        if (getResources().getBoolean(R.bool.twoPaneMode)) {
            // all good, we use the fragments defined in the layout
            return;
        }
        // if savedInstanceState is null we do some cleanup
        if (savedInstanceState != null) { (1)
            // cleanup any existing fragments in case we are in detailed mode
            getFragmentManager().executePendingTransactions();
            Fragment fragmentById = getFragmentManager().
                findFragmentById(R.id.fragment_container);
            if (fragmentById!=null) {
                getFragmentManager().beginTransaction()
                    .remove(fragmentById).commit();
            }
        }
        MyListFragment listFragment = new MyListFragment();
        FrameLayout viewById = (FrameLayout) findViewById(R.id.fragment_container);
        getFragmentManager().beginTransaction()
            .replace(R.id.fragment_container, listFragment).commit();
    }
    @Override
    public void onRssItemSelected(String link) {
        if (getResources().getBoolean(R.bool.twoPaneMode)) {
            DetailFragment fragment = (DetailFragment) getFragmentManager()
                .findFragmentById(R.id.detailFragment);
            fragment.setText(link);
        } else {
            // replace the fragment
            // Create fragment and give it an argument for the selected article
            DetailFragment newFragment = new DetailFragment();
            Bundle args = new Bundle();
            args.putString(DetailFragment.EXTRA_URL, link);
            newFragment.setArguments(args);
            FragmentTransaction transaction = getFragmentManager().beginTransaction();
            // Replace whatever is in the fragment_container view with this fragment,
            // and add the transaction to the back stack so the user can navigate back
            transaction.replace(R.id.fragment_container, newFragment);
            transaction.addToBackStack(null);
            // Commit the transaction
            transaction.commit();
        }
    }
}

```

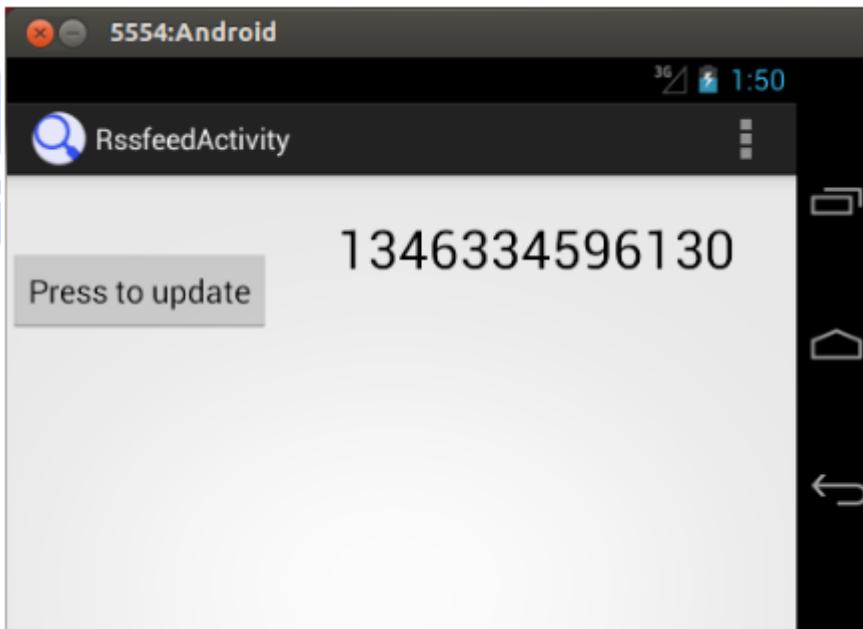
```
{
}
```

این فرایند پاک سازی ضروری است، چراکه اپلیکیشن ما به صورت dynamic و در زمان اجرا بین دو حالت تک قطعه یا نمایش یک fragment در لحظه و نمایش دو قطعه یا fragment در آن واحد، تغییر وضعیت می دهد. البته سناریوی حاضر غیرمعمول بوده و معمولاً انتخاب یک یا دو fragment برای اپلیکیشن بستگی به حداقل عرض نمایشگر دارد و این پیکربندی در زمان اجرای برنامه تغییر نمی یابد. fragment سعی دارد با cache یا ذخیره ی موقتیFragmentManager کارایی را افزایش دهد، از اینرو لازم است detailed fragment موجود را حذف نمایید.

## تست اپلیکیشن و اعتبارسنجی رفتارهای آن

اپلیکیشن خود را اجرا نمایید. اکنون اگر اپلیکیشن خود را در نمای عمودی (portrait mode) اجرا نمایید، طبیعتاً باید در لحظه تنها یک fragment در نمایشگر قابل مشاهده باشد. در نمای افقی می بایست هر دو fragment همزمان برای کاربر نمایش داده شود.

جهت یا وضعیت نمایش شبیه ساز را تغییر دهید. دکمه را در دو حالت نمای عمودی و افقی فشار داده و مطمئن شوید که detail activity زمان حاضر را به درستی نمایش می دهد.





بخش دوم :

## استفاده از نوارابزار در اپلیکیشن های اندروید/Action bar/Android

این مبحث تمرکز خود را بر روی آموزش نحوه ی پیاده سازی و استفاده از toolbar در اپلیکیشن های اندرویدی قرار می دهد. آموزش حاضر همچنین چگونگی استفاده از widget/کامپوننت رابط کاربری toolbar را برای شما شرح می دهد. این آموزش مبتنی بر ویرایش 6.0 سیستم عامل اندروید می باشد (= API یا کتابخانه های اندروید ورژن 23).

### 6-3- شرح مفهوم Toolbar

## 3-6-1 Toolbar در اندروید چیست و چه کاربردی دارد؟ (action bar)

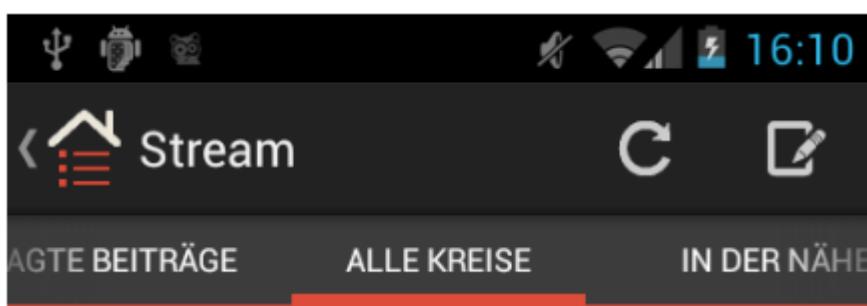
در ویرایش 5.0 سیستم عامل اندروید جای toolbar در اندروید 4 را گرفت تا محدودیت های جاری را برطرف ساخته و راه حلی نوین برای طراحی اپلیکیشن را در اختیار برنامه نویس قرار دهد. این widget یا کامپوننت UI از طریق view group به نام Toolbar در دسترس توسعه دهنده قرار می گیرد.

می توان toolbar را به راحتی در فایل layout جایگذاری نمود. این کامپوننت UI قادر است عنوان activity، آیکون، عملیاتی که قابل فعال سازی هستند (action ها) و حتی view ها و سایر آیتم های تعاملی را دربرگیرد. مورد دیگر استفاده ای toolbar در پیمایش و راهبری بین بخش های مختلف اپلیکیشن است.

قبل از ویرایش 5.0 اندروید، مکان قرار گیری نوار ابزار (همان actionBar در ویرایش های قبلی) به صورت ثابت در بالای صفحه یا activity بوده (در آنجا hard code شده بود).

اکنون این امکان وجود دارد که نوار ابزار را در پوسته ای مورد استفاده (theme) غیرفعال نمود. کامپوننت UI نام برده در تمامی پوسته های درون ساخته ای اندروید به صورت پیش فرض فعال می باشد.

تصویر زیر نوار ابزار اپلیکیشن اندرویدی Google+ را به همراه آیتم های تعاملی و نوار راهبری (navigation) نشان می دهد. در بالای آن سمت چپ، علامتی قابل مشاهده است که کاربر می تواند با کلیک بر روی آن نوار راهبری را باز کرده و در اپلیکیشن پیمایش کند.



## ActionBar-3-6-2 بر روی دستگاه هایی که ورژن کتابخانه های اندروید

مورد استفاده در آن ها پایین تر از 21 است (API 21)

Toolbar در ویرایش 5.0 اندروید (ورژن کتابخانه های اندروید 21) برای اولین بار معرفی شد و در واقع جایگزینی برای actionBar در ورژن های قبلی این سیستم عامل گردید. اگر می خواهید از این کاپوننت در ویرایش های قدیمی تر اندروید استفاده نمایید، در آن صورت کافی است از downport ای که توسط appcompat-v7 در اختیار توسعه دهنده قرار می گیرد، استفاده نمایید.

جهت استفاده از toolbar در چنین دستگاه هایی، دستوری مانند نمونه‌ی زیر را به فایل build خود اضافه نمایید:

```
compile 'com.android.support:appcompat-v7:22.2.0'
```

برای آموزش نحوه نصب library v7 به آدرس

<http://developer.android.com/tools/support-library/setup.html> مراجعه نمایید.

## Options menu-3-6-3

اپلیکیشن هایی که target SDK آن ها بر روی پایین تر از API 11 تنظیم شده، از options menu استفاده می کنند. البته اگر چنین دکمه ای در دستگاه موجود باشد. این کامپوننت UI با کلیک کاربر بر روی دکمه‌ی Options به نمایش در می آید. Toolbar نسبت به options menu کاربر پسندتر است زیرا action bar همیشه قابل مشاهده و در دسترس است. در حالی که options menu فقط در صورت درخواست نمایان می شود. در واقع options menu یک کاستی برجسته دارد: ممکن است کاربر متوجه وجود گزینه ها و تنظیمات ارائه شده توسط اپلیکیشن نشود.

## Toolbar-7-3 استفاده از

### 3-7-3-تعریف آیتم های نوار ابزار/ toolbar actions در

المان هایی که در نوارابزار مشاهده می کنید در اندروید actions خوانده می شوند. اگرچه می توان آیتم های نوار ابزار را با کدنویسی تعریف کرد، با این حال توسعه دهندهان معمولاً آن ها را در یک فایل محتوا XML تحت پوشه res/ تعریف می کنند.

تعریف منوها در یک فایل مجزا تحت پوشه res/menu ذخیره می شوند. ابزار اندروید خود به صورت خودکار یک اشاره گر به آیتم های منو جهت دسترسی به منابع آن در فایل R ایجاد می کند.

یک معمولاً entry معمولاً activity را از طریق متده onCreateOptionsMenu() به نوارابزار تزریق می کند.

خصیصه showAsAction (attribute) به شما این اجازه را می دهد تا نحوه نمایش و ظاهر آیتم مورد نظر در نوارابزار را تعیین نمایید. به عنوان مثال، خصیصه ifRoom مشخص می کند آیا فضای کافی برای نمایش المان وجود دارد یا خیر.

```
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
<item
    android:id="@+id/action_refresh"
    android:orderInCategory="100"
    android:showAsAction="always"
    android:icon="@drawable/ic_action_refresh"
    android:title="Refresh"/>
<item
    android:id="@+id/action_settings"
    android:title="Settings">
</item>
</menu>
```

کلاس MenuInflater این امکان را فراهم می آورد تا action های تعریف شده در فایل XML را نموده (آن ها را جهت قرار گرفتن در view به آبجکت activity تبدیل کرده) سپس آن ها را به اضافه نمایید. می توانید با فراخوانی getMenuInflater() در سطح action bar این دسترسی داشته باشید. نمونه کد زیر نحوه پیاده سازی و ایجاد action ها را نمایش می دهد.

```

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.mainmenu, menu);
    return true;
}

```

**نکته:** اگرچه می توان action ها را در کد برنامه نیز تعریف کرد، با این حال توصیه می شود این کار را در فایل های XML انجام دهید چرا که میزان کدهای تکراری (boilerplate code) را کاهش می دهد.

## 2-7-3-عملیاتی که در پی انتخاب آیتم های نوار ابزار رخ می دهند (واکنش نشان دادن به انتخاب action ها)

زمانی که action ای کلیک می شود، متد onOptionsItemSelected() در activity مربوطه فراخوانی می گردد. این متد action انتخابی را به عنوان پارامتر ورودی دریافت می کند. کاربرد این متد در نمونه‌ی زیر به نمایش گذاشته شده است.

```

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        // action with ID action_refresh was selected
        case R.id.action_refresh:
            Toast.makeText(this, "Refresh selected", Toast.LENGTH_SHORT)
                .show();
            break;
        // action with ID action_settings was selected
        case R.id.action_settings:
            Toast.makeText(this, "Settings selected", Toast.LENGTH_SHORT)
                .show();
            break;
        default:
            break;
    }
    return true;
}

```

### 3-7-3-جستجو برای یک action bar یا آیتم در نوارابزار/

به منظور جستجو برای یک action یا آیتم در منو، متدهای `findItem()` از کلاس `Menu` را فراخوانی نمایید. به وسیله ای این متدهای می توانید با استفاده از `id` یک آیتم به آن دسترسی داشته باشید.

### 3-7-4-ویرایش منو

متدهای `onCreateOptionsMenu()` تنها یکبار صدا خورده می شود. در صورت نیاز به ویرایش منو در آینده، لازم است متدهای `invalidateOptionsMenu()` را فراخوانی نمایید. پس از آن تابع `onCreateOptionsMenu()` بار دیگر صدا زده می شود.

### 3-7-5-وابسته کردن toolbar و Contextual action mode

#### (contextual action mode)

به قرائت با استفاده از `Contextual action mode` یک نوارابزار (وابسته به قرائت و بستر جاری) است که برای انجام `subtask` ها و وظایف کوچک برای مدت زمان معینی، به صورت موقتی بر روی نوارابزار اصلی اپلیکیشن ظاهر می شود.

معمولًا با انتخاب یک آیتم یا کلیک طولانی مدت بر روی آن ظاهر می شود.

به منظور پیاده سازی این نوع `toolbar`، کافی است متدهای `startActionMode()` و `endActionMode()` را بر روی یک `Activity` در فراخوانی نمایید. متدهای `startActionMode()` و `endActionMode()` از `ActionMode.Callback` ارث می کنند. این آبجکت وظیفه ای مدیریت چرخه ای `Contextual action bar` را بر عهده دارد. همچنین می توانید با استفاده از متدهای `registerForContextMenu(View)`، یک منوی وابسته به قرائت/Context menu نیز با کلیک طولانی مدت بر روی `View` مورد نظر اختصاص دهید.

هر بار که یک `Context menu` فعال می شود، متدهای `onCreateContextMenu()` و `onContextItemSelected()` را اجرا می کند.

هر بار که یک `Context menu` فعال می شود، متدهای `onCreateContextMenu()` و `onContextItemSelected()` را اجرا می کند. می دانید چرا؟ `Context menu` پس از هر بار استفاده کاملا دور اندخته می شود.

توصیه می شود تا حد امکان بجای `Context menu` از `Contextual action mode` استفاده نمایید.

### ۷-۳-۱- اضافه نمودن آیتم به action bar با استفاده از fragment ها

Fragment ها نیز می توانند آیتم و المان هایی را به نوارابزار یا toolbar اپلیکیشن اضافه کنند. برای نیل به این هدف، می بایست متدهای `setHasOptionsMenu(true)` را در متدهای `onCreate()` و `onCreateOptionsMenu()` از framework یا چارچوب نرم افزاری Android در این شرایط مورد نظر فراخوانی نمایید. در این شرایط fragment می تواند آیتم های منو را به toolbar اضافه کند. در این حالت `onCreateOptionsMenu()` را در کلاس fragment فراخوانی می کند. در این حالت `onCreateOptionsMenu()` می تواند آیتم های منو را به toolbar اضافه کند.

### ۷-۳-۲- تنظیم قابلیت رویت (visibility) و دسترسی toolbar

می توانید قابلیت رویت و visibility نوارابزار را در زمان اجرای برنامه تنظیم نمایید. کد زیر نحوه انجام این کار را به نمایش می گذارد.

```
ActionBar actionBar = getActionBar();
actionBar.hide();
// more stuff here...
actionBar.show();
```

شما می توانید متنهای `actionBar.hide()` و `actionBar.show()` را در زمان اجرای برنامه (runtime) نمایش داده می شود را ویرایش نمایید. تکه کد زیر نحوه انجام این کار را نمایش می دهد.

```
ActionBar actionBar = getActionBar();
actionBar.setSubTitle("mytest");
actionBar.setTitle("vogella.com");
```

### ۷-۳-۳- تخصیص یک عکس به action bar

این امکان نیز برای شما وجود دارد که یک Drawable را از طریق متدهای `setBackgroundDrawable()` و `setBackground()` نوارابزار خود تنظیم نمایید.

Toolbar اپلیکیشن خود عکس را مقیاس دهی کرده و با توجه فضای جاری اندازه بندی می کند. از اینرو بهتر است که یک فایل drawable انعطاف پذیر و قابل تنظیم فراهم می کنید. برای مثال می توانید یک عکس XML یا یک عکس 9-patch را به عنوان پس زمینه `setBackground()` یا `setBackgroundDrawable()` نمایید.

از ویرایش 4.2 سیستم عامل اندروید، پس زمینه ای action bar را نیز می توان با استفاده از diming navigation یا کم رنگ کردن دکمه ای پیمایش (button) متحرک یا پویا نمایی کرد.

### 7-3-پنهان یا کم رنگ کردن دکمه ای پیمایش (button)

شما می توانید دکمه ای پیمایش نرم افزاری مورد نظر را در اپلیکیشن اندرویدی خود پنهان نموده و بدین وسیله فضای بیشتری در دسترس داشته باشید. زمانی که کاربر بر روی دکمه ای صفحه کلیک می کند، دکمه ای پیمایش به صورت خودکار نمایان می شود.

کد لازم برای پیاده سازی این قابلیت در زیر به نمایش گذاشته شده است.

```
getWindow().  
getDecorView().  
setSystemUiVisibility(View.SYSTEM_UI_FLAG_HIDE_NAVIGATION);
```

تصاویر زیر یک اپلیکیشن را با/بدون دکمه های پیمایش نمایش می دهد.





317

آدرس آموزشگاه : تهران - خیابان شریعتی - بالاتر از خیابان ملک - جنب بانک صادرات - پلاک 651 طبقه دوم - واحد 7

88146323 - 88446780 - 88146330



### 10-7-3- استفاده از حالت نمایش تمام صفحه (immersive full screen mode)

از ویرایش 4.4 (ورژن کتابخانه های اندروید یا API 19) به بعد، شما می توانید اپلیکیشن خود را در حالت نمایش تمام صفحه قرار دهید (immersive full screen mode). اولین باری که این اتفاق می افتد، سیستم اندروید اطلاعاتی را به کاربر نشان می دهد. این اطلاعات حاکی از این است که

کاربر می تواند با حرکت downward swipe (لمس سطح نمایشگر به سمت پایین در ناحیه ای که نوارابزار سیستم معمولاً قابل مشاهده می باشد) اپلیکیشن را به حالت قبلی بازگردانده و system bar ها را بار دیگر به نمایش بگذارد.

به عنوان مثال، متده زیر activity یا صفحه ای جاری اپلیکیشن را در حالت تمام صفحه قرار می دهد.

```
// This method hides the system bars and resize the content
private void hideSystemUI() {
    getWindow().getDecorView().setSystemUiVisibility(
        View.SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN
            | View.SYSTEM_UI_FLAG_HIDE_NAVIGATION // hide nav bar
            | View.SYSTEM_UI_FLAG_FULLSCREEN // hide status bar
            //remove the following flag for version < API 19
            | View.SYSTEM_UI_FLAG_IMMERSIVE
    );
}
```

**۷-۳-۱۱-۳-پیاده سازی قابلیت دو نیم کردن نوار ابزار (split toolbar)**  
می توانید به سیستم اعلان نمایید که در صورت عدم وجود فضای کافی، نوارابزار را به صورت خودکار به دو نیم تقسیم کند. جهت فعال سازی این قابلیت می توانید پارامتر android:uiOptions="SplitActionBarWhenNarrow" را در تعریف activity در `AndroidManifest.xml` لحظ نمایید.

**توجه:** با فعال سازی گزینه `SplitActionBarWhenNarrow`، اندروید می تواند نوار ابزار را به دو بخش تقسیم کند. سپس سیستم بر اساس فضای جاری و در زمان اجرای برنامه تصمیم می گیرد آیا نوارابزار را به دو بخش تقسیم کند یا خیر.

## تمرین: استفاده از Contextual action mode در اپلیکیشن اندرویدی خود

### هدف

در تمرین جاری، شما یک contextual action mode به اپلیکیشن خود اضافه خواهید نمود.

## 7-3-ایجاد محتوا و منابع مربوطه menu resource در پوشه res/menu

برای این منظور، یک فایل XML حامل محتوای منو (xml resource menu) با نام `menu.xml` باز نمایید. برای جزئیات بیشتر به `<android:actionModeMenu>` مراجعه کنید.

```

<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/action_share"
        android:title="Share it"
    >
    </item>
</menu>

```

## 7-3- تنظیم و ویرایش کد برنامه

ابتدا interface مربوط به fragment را به صورت زیر ارث بری نمایید.

```

public class MyListFragment extends Fragment {
    public void goToActionMode(RssItem item) {
        listener.goToActionMode(item);
    }
    public interface OnItemSelectedListener {
        public void onRssItemSelected(String link);
        public void goToActionMode(RssItem item);
    }
}

```

کلاس activity خود را جهت پیاده سازی متد جدید و ActionMode.Callback به صورت زیر ویرایش نمایید.

```

package com.example.android.rssreader;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.ActionMode;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.widget.Toast;
import android.widget.Toolbar;
import com.example.android.rssfeedlibrary.RssItem;
public class RssfeedActivity extends AppCompatActivity
    implements MyListFragment.OnItemSelectedListener,
    ActionMode.Callback {
    private RssItem selectedRssItem;
}

```

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Toolbar tb = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(tb);
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    Toolbar tb = (Toolbar) findViewById(R.id.toolbar);
    tb.inflateMenu(R.menu.mainmenu);
    tb.setOnMenuItemClickListener(
        new Toolbar.OnMenuItemClickListener() {
            @Override
            public boolean onMenuItemClick(MenuItem item) {
                return onOptionsItemSelected(item);
            }
        });
    return true;
}
//NEW
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.action_refresh:
            MyListFragment fragment = (MyListFragment) getSupportFragmentManager()
                .findFragmentById(R.id.listFragment);
            fragment.updateListContent();
            break;
        case R.id.action_settings:
            Intent intent = new Intent(this, MyPreferences.class);
            startActivity(intent);
            Toast.makeText(this, "Action Settings selected", Toast.LENGTH_SHORT)
                .show();
            break;
        default:
            break;
    }
    return true;
}
@Override
public void onRssItemSelected(String link) {
    if (getResources().getBoolean(R.bool.twoPaneMode)) {
        DetailFragment fragment = (DetailFragment) getSupportFragmentManager()
            .findFragmentById(R.id.detailFragment);
        fragment.setText(link);
    } else {
        Intent intent = new Intent(getApplicationContext(),
            DetailActivity.class);
        intent.putExtra(DetailActivity.EXTRA_URL, link);
        startActivity(intent);
    }
}

```

```

    }

    @Override
    public void showContextMenu(RssItem item) {
        this.selectedRssItem = item;
        startActionMode(this);
    }

    @Override
    public void goToActionMode(RssItem item) {
        this.selectedRssItem = item;
        startActionMode(this);
    }

    @Override
    public boolean onCreateActionMode(ActionMode mode, Menu menu) {
        MenuInflater inflater = mode.getMenuInflater();
        inflater.inflate(R.menu.actionmode, menu);
        return true;
    }

    @Override
    public boolean onPrepareActionMode(ActionMode mode, Menu menu) {
        return false;
    }

    @Override
    public boolean onActionItemClicked(ActionMode mode, MenuItem item) {
        Intent intent = new Intent(Intent.ACTION_SEND);
        intent.putExtra(Intent.EXTRA_TEXT, "I found this interesting link" +
            selectedRssItem.getLink());
        intent.setType("text/plain");
        startActivity(intent);
        mode.finish(); // Action picked, so close the CAB
        selectedRssItem = null;
        return true;
    }

    @Override
    public void onDestroyActionMode(ActionMode mode) {
    }
}

```

در کلاس adapter خود، یک `LongClickListener` پیاده سازی نمایید. با پیاده سازی `contextual action mode` مزبور، فعال و راه اندازی می شود.

```

package com.example.android.rssreader;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;
import com.bumptech.glide.Glide;
import com.example.android.rssfeedlibrary.RssItem;
import java.util.List;
import java.util.Random;

```

```

public class RssItemAdapter
    extends RecyclerView.Adapter<RssItemAdapter.ViewHolder> {
private List<RssItem> rssItems;
private MyListFragment myListFragment;
@Override
public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
    View v= null;
    v = LayoutInflater.
        from(parent.getContext()).
        inflate(R.layout.rowlayout, parent, false);
    return new ViewHolder(v);
}
public static class ViewHolder extends RecyclerView.ViewHolder {
    public View mainLayout;
    public TextView txtHeader;
    public TextView txtFooter;
    public ImageView imageView;
    public ViewHolder(View v) {
        super(v);
        mainLayout = v;
        txtHeader = (TextView) v.findViewById(R.id.rsstitle);
        txtFooter = (TextView) v.findViewById(R.id.rssurl);
        imageView = (ImageView) v.findViewById(R.id.icon);
    }
}
@Override
public void onBindViewHolder(final ViewHolder holder, final int position) {
    final RssItem rssItem = rssItems.get(position);
    holder.txtHeader.setText(rssItem.getTitle());
    holder.txtFooter.setText(rssItem.getLink());
    holder.mainLayout.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            myListFragment.updateDetail(rssItem.getLink());
        }
    });
    holder.mainLayout.setOnLongClickListener(new View.OnLongClickListener() {
        @Override
        public boolean onLongClick(View v) {
            myListFragment.goToActionMode(rssItem);
            return true;
        }
    });
}
@Override
public int getItemCount() {
    return rssItems.size();
}
public RssItemAdapter(List<RssItem> rssItems, MyListFragment myListFragment) {
    this.rssItems = rssItems;
    this.myListFragment = myListFragment;
}

```

}

## action bar Dynamic-8-3 تعریف کردن

### View-3-8-1 action bar هایی با پیاده سازی اختصاصی در

می توانید یک view اختصاصی به action bar اضافه نمایید. برای مثال، یک button یا text field می توانید در نوار ابزار اپلیکیشن خود داشته باشید.

برای این منظور، لازم است متدهای setCustomView از کلاس ActionView را بکار ببرید. سپس با استفاده از فراخوانی setDisplayOptions() و ارسال پارامتر toolbar.DISPLAY\_SHOW\_CUSTOM می توانید view های اختصاصی در action bar را فعال سازی نمایید.

به طور مثال، می توانید یک فایل layout تعریف کنید که در بردارنده ی المان EditText می باشد.

```
<?xml version="1.0" encoding="utf-8"?>
<EditText xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/searchfield"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:inputType="textFilter" >
</EditText>
```

این layout را می توان به وسیله ی کد زیر به activity در action bar مورد نظر متصل کرد. تکه کد زیر علاوه بر متصل کردن layout، یک گوش فراخوان یا listener custom view نیز به الحاق می کند.

```
package com.vogella.android.actionbar.customviews;
import android.app.ActionBar;
import android.app.Activity;
import android.os.Bundle;
import android.view.KeyEvent;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.TextView.OnEditorActionListener;
import android.widget.Toast;
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
```

```

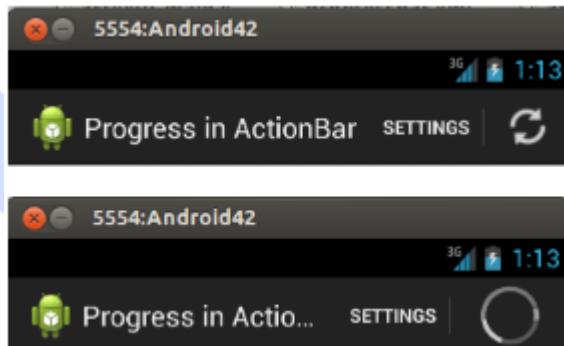
ActionBar actionBar = getActionBar();
// add the custom view to the action bar
actionBar.setCustomView(R.layout.actionbar_view);
EditText search = (EditText) actionBar.getCustomView().findViewById(
    R.id.searchfield);
search.setOnEditorActionListener(new OnEditorActionListener() {
    @Override
    public boolean onEditorAction(TextView v, int actionId,
        KeyEvent event) {
        Toast.makeText(MainActivity.this, "Search triggered",
        Toast.LENGTH_LONG).show();
        return false;
    }
});
actionBar.setDisplayOptions(ActionBar.DISPLAY_SHOW_CUSTOM
    | ActionBar.DISPLAY_SHOW_HOME);
}
}

```

### Action view-3-8-2

یک کامپوننت UI یا widget هست که بجای دکمه ی آیتم (action) در نوارابزار نمایش داده می شود. با استفاده از این ویژگی می توانید یک آیتم (action) را با آبجکت Progressbar یا هر view مجاز دیگری جایگزین نمایید. به منظور تعریف یک action view جایگزین برای آیتم (action) در toolbar، می توانید خصیصه (attribute) android:actionLayout (attribute) را جهت ایجاد فایل layout و خصیصه ی android:actionViewClass مورد استفاده قرار دهید.

این جایگزینی در تصاویر زیر به نمایش گذاشته شده است.



ProgressBar زیر آیکون را در زمان اجرا برنامه با ای که حامل یک آبجکت action view می باشد، جایگزین می نماید.

```

package com.vogella.android.actionbar.progress;
import android.app.ActionBar;
import android.app.Activity;
import android.os.AsyncTask;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
public class MainActivity extends Activity {
    private MenuItem menuItem;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ActionBar actionBar = getActionBar();
        actionBar.setDisplayOptions(ActionBar.DISPLAY_SHOW_HOME
                | ActionBar.DISPLAY_SHOW_TITLE | ActionBar.DISPLAY_SHOW_CUSTOM);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
        case R.id.menu_load:
            menuItem = item;
            menuItem.setActionView(R.layout.progressbar);
            menuItem.expandActionView();
            TestTask task = new TestTask();
            task.execute("test");
            break;
        default:
            break;
        }
        return true;
    }
    private class TestTask extends AsyncTask<String, Void, String> {
        @Override
        protected String doInBackground(String... params) {
            // Simulate something long running
            try {
                Thread.sleep(2000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            return null;
        }
        @Override
        protected void onPostExecute(String result) {
            menuItem.collapseActionView();
            menuItem.setActionView(null);
        }
    }
}

```

```
    }  
};  
}
```

کد زیر، محتوای layout مورد استفاده ای action view را نشان می دهد.

```
<?xml version="1.0" encoding="utf-8"?>
<ProgressBar xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/progressBar2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">
</ProgressBar>
```

کد زیر فایل های XML که برای منو استفاده می شود را نشان می دهد.

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">  
    <item  
        android:id="@+id/menu_settings"  
        android:orderInCategory="100"  
        android:showAsAction="always"  
        android:title="Settings"  
    />  
    <item  
        android:id="@+id/menu_load"  
        android:icon="@drawable/navigation_refresh"  
        android:orderInCategory="200"  
        android:showAsAction="always"  
        android:title="Load"/>  
</menu>
```

## Action provider-9-3

**Action provider-3-9-1** چیست و چه کاربردی دارد؟

در واقع همان Action provider است که layout اختصاصی خود را دارد. Action provider ممکن است به صورت یک button یا آیتم در منو ظاهر شود، اما زمانی که کاربر بر روی action کلیک می کند، action provider رفتار action را بر اساس تعریف شما، کنترل و هدایت می کند. برای مثال، action bar می تواند با نمایش یک منو به کلیک کاربر واکنش نشان دهد. به عبارت دیگر، action provider به شما امکان می دهد تا قابلیت تعامل و رفتار پیچیده را در قالب تنها یک action به راحتی پیاده سازی نمایید. این action قادر است action view هایی تولید کند که در action bar استفاده می شوند، زیرمنوها را به صورت پویا و در زمان اجرای اپلیکیشن با داده پر کند و فراخوانی های پیش فرض action را مدیریت نماید.

کلاسی که action bar از آن مشتق می شود، کلاس پایه/پدر ActionProvider می باشد.

در حال حاضر، محیط اندروید (Android platform) دو action provider دهنگان قرار می دهد: 1. ShareActionProvider .2 MediaRouteActionProvider

## تمرین: استفاده از ShareActionProvider

تمرین زیر استفاده از ShareActionProvider را نمایش می دهد. این action provider به شما اجازه می دهد تا محتوای انتخابی را از اپلیکیشن هایی که Intent.ACTION\_SEND را ثبت کرده اند، دریافت نمایید.

جهت استفاده از ShareActionProvider، می بایست یک آیتم در منو ویژه‌ی آن تعریف کرده و intent ای به آن تخصیص دهید که داده های مورد نظر برای اشتراک را دربرمی گیرد.

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
    <item android:id="@+id/menu_share"
        android:title="Share"
        android:showAsAction="ifRoom"
        android:actionProviderClass="android.widget.ShareActionProvider" />
    <item
        android:id="@+id/item1"
        android:showAsAction="ifRoom"
        android:title="More entries...">
    </item>
</menu>
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.activity_main, menu);
    // Get the ActionProvider for later usage
    provider = (ShareActionProvider) menu.findItem(R.id.menu_share)
        .getActionProvider();
    return true;
}
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.menu_share:
            doShare();
            break;
        default:
            break;
    }
    return true;
}
```

```

    }
    public void doShare() {
        // populate the share intent with data
        Intent intent = new Intent(Intent.ACTION_SEND);
        intent.setType("text/plain");
        intent.putExtra(Intent.EXTRA_TEXT, "This is a message for you");
        provider.setShareIntent(intent);
    }
}

```

## 10-3-پیمایش در اپلیکیشن از طریق آیکون

### 10-3-استفاده از آیکون اپلیکیشن در action bar جهت راهبری به صفحه اصلی

در action bar آیکونی از اپلیکیشن خود را مشاهده می کنید. آیتم مزبور home icon نام داشته و شما می بایست یک action به آن اختصاص دهید. بهتر است کاربر با کلیک بر روی آیکون، به صفحه ای اصلی برنامه (main activity) هدایت شود.

پس از انتخاب شدن action مورد نظر، متده onOptionsItemSelected() همراه با آیتم یا ای صدا خورده می شود که شناسه یا ID آن android.R.id.home باشد.

پیش از ویرایش 4.1 اندروید، توسعه دهنده می بایست شناسه android.R.id.home را در متده onOptionMenuItemSelected() بکار می برد و امکان انتخاب دکمه ی home را فعال سازی SecondActivity در کد زیر به نمایش گذاشته شده است. آیکون home در MainActivity کاربر را به SecondActivity در اپلیکیشن هدایت می کند.

```

package com.vogella.android.actionbar.homebutton;
import android.os.Bundle;
import android.app.ActionBar;
import android.app.Activity;
import android.content.Intent;
import android.view.Menu;
import android.view.MenuItem;

```

```

public class SecondActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // enable the home button
        ActionBar actionBar = getActionBar();
        actionBar.setHomeButtonEnabled(true);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case android.R.id.home:
                Intent intent = new Intent(this, MainActivity.class);
                intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                startActivity(intent);
                break;
            // Something else
            case R.id.action_settings:
                intent = new Intent(this, ThirdActivity.class);
                startActivity(intent);
            default:
                break;
        }
        return super.onOptionsItemSelected(item);
    }
}

```

**نکته:** از Android 4.1 به بعد، برنامه نویس می تواند با برابر قرار دادن مقدار `activity` با نام `activity` پدر (برای مثال "MainActivity") در فایل تنظیمات اپلیکیشن (`AndroidManifest.xml`، به راحتی کاربر را از `activity` دوم به `activity` اصلی هدایت کرده و تعیین کند که کاربر با کلیک بر روی آیکون `home` به `activity` اصلی برنامه ارجاع داده شود.

```

<activity
    android:name="SecondActivity"
    android:label="@string/app_name"
    android:parentActivityName="MainActivity" >
</activity>

```

### 10-3- استفاده از آیکون اپلیکیشن به عنوان دکمه Up

می توانید از آیکون اپلیکیشن به عنوان دکمه Up استفاده نمایید به طوری که کاربر با کلیک بر روی آن به میزبان که پدر activity کنونی هست، هدایت شود. دکمه Up در دستگاه همیشه کاربر را به activity یا صفحه ای قبلی بازگشت می دهد.

رفتار و کاربرد هر دو دکمه را می توان متفاوت تعریف کرد، برای مثال زمانی که کاربر آپشنی را فعال کند که به او اجازه ای نوشتن ایمیل از صفحه ای اصلی را می دهد، برنامه را گونه ای تعریف کرده باشید که کلیک روی دکمه Up کاربر را به صفحه ای اصلی و کلیک بر روی دکمه Up کاربر را به activity ای که نمایی مختصر از تمامی ایمیل ها ارائه می دهد، هدایت نماید.

به منظور فعال سازی Up، می توانید تکه کد زیر را در activity خود اضافه نمایید.

```
actionBar.setDisplayUseLogoEnabled(false);
actionBar.setDisplayHomeAsUpEnabled(true);
```

**نکته:** درج تکه کد بالا در کلاس activity صرفا Up را بر روی آیکون home فعال می سازد. شما بایستی رفتار لازم را در کلاس activity خود و داخل متده onOptionsItemSelected() پیاده سازی نمایید. action مربوطه همچنان با همان شناسه `android.R.id.home` فراخوانی می شود.

### تمرین: پیاده سازی ActionBar

پروژه

در این بخش action و آیتم هایی تعاملی در action bar ایجاد خواهید نمود که به انتخاب کاربر واکنش نشان می دهند.

تمرینی که در بخش حاضر دنبال می کنیم مبتنی بر آموزش fragment است. اگر قبلا پروژه ای آن را نوشته اید می توانید همان را دوباره استفاده کنید.

### تمرین: اضافه کردن toolbar به اپلیکیشن

## افزودن آیکون refresh در نوار ابزار

پروژه‌ی RSS Reader را ادامه دهید. با طی نمودن میسر زیر یک drawable به نام ic\_refresh را ایجاد نمایید: File ▶ New ▶ Vector Asset

### 3-10-3- اضافه کردن منابع لازم برای منو

یک فایل XML جدید به نام mainmenu.xml ایجاد نمایید. سپس یک آیتم به فایل مورد نظر اضافه کنید، طوری که فایل XML نهایی ظاهری مشابه زیر داشته باشد. می بایست این آیتم‌ها را به صورت دستی تایپ کنید چرا که محیط کاری Android Studio تا زمان نگارش آموزش حاضر، این قابلیت را ندارد که کدهای ساخت منوها را به صورت خودکار برای توسعه دهنده تکمیل کند.

```
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
    <item
        android:id="@+id/action_settings"
        android:title="Settings"
        android:showAsAction="never"
    >
    </item>
</menu>
```

یک فایل XML جدید به نام listfragment\_menu.xml برای منوی اپلیکیشن خود تولید نمایید. سپس یک آیتم مانند زیر به فایل اضافه کنید تا فایل XML نهایی ظاهری مشابه زیر داشته باشد. می بایست این آیتم‌ها را خود به صورت دستی بنویسید چرا که تا زمان نگارش آموزش حاضر، محیط کاری Android Studio قابلیت تکمیل خودکار کدها را ندارد.

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:app="http://schemas.android.com/apk/res-auto"
      xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:title="Refresh"
          android:icon="@drawable/ic_refresh"
          android:id="@+id/action_refresh"
          android:enabled="false"
          android:showAsAction="ifRoom" />
</menu>
```

### 4-3-غیرفعال کردن action bar پیش فرض از طریق تگ style

از آنجایی که برای پروژه از toolbar اختصاصی خود استفاده می کنید، لازم است action bar را در فایل res/values/styles.xml غیر فعال نمایید.

```
<resources>
    <!-- Base application theme. -->
    <style name="AppTheme" parent="android:Theme.Material.Light.NoActionBar">
        <!-- Customize your theme here. -->
    </style>
</resources>
```

### 4-3-اضافه کردن یک آبجکت toolbar (view)

فایل های layout مربوط به RssfeedActivity خود را طوری ویرایش کنید که آیتم (entry) زیر را دربرداشته باشد.

```
<?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"
    tools:context=".MainActivity"
    android:orientation="vertical">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:orientation="horizontal"
        android:layout_weight="1"
        android:paddingBottom="@dimen/activity_vertical_margin"
        android:paddingLeft="@dimen/activity_horizontal_margin"
        android:paddingRight="@dimen/activity_horizontal_margin"
        android:paddingTop="@dimen/activity_vertical_margin">
    </LinearLayout>
    <fragment
        android:id="@+id/listFragment"
        class="com.example.android.rssreader.MyListFragment"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1"          tools:layout="@layout/fragment_rsslist_overview"></fragment>
    <fragment
        android:id="@+id/detailFragment"
        class="com.example.android.rssreader.DetailFragment"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="2"
        tools:layout="@layout/fragment_rssitem_detail" />
</LinearLayout>
```

```
<Toolbar
    android:id="@+id/toolbar"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    >
</Toolbar>
</LinearLayout>
```

همین کار را برای نمای عمودی (portrait mode) نیز انجام دهید.

```
<?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"
    tools:context=".RssfeedActivity"
    android:orientation="vertical">
    <FrameLayout
        android:id="@+id/fragment_container"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:orientation="horizontal"
        android:paddingBottom="@dimen/activity_vertical_margin"
        android:paddingLeft="@dimen/activity_horizontal_margin"
        android:paddingRight="@dimen/activity_horizontal_margin"
        android:paddingTop="@dimen/activity_vertical_margin"
        >
    </FrameLayout>
    <Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        >
    </Toolbar>
</LinearLayout>
```

بدنه ی کلاس RssfeedActivity خود را جهت تنظیم آبجکت toolbar مانند زیر ویرایش نمایید.

```
package com.example.android.rssreader;
import android.app.Activity;
import android.app.Fragment;
import android.app.FragmentManager;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.FrameLayout;
import android.widget.Toast;
import android.widget.Toolbar;
public class RssfeedActivity extends Activity implements
    MyListFragment.OnItemSelectedListener {
    // as before
    @Override
```

```

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_rssfeed);
    //
    Toolbar tb = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(tb);
    // more code as before
}

//  

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    Toolbar tb = (Toolbar) findViewById(R.id.toolbar);
    tb.inflateMenu(R.menu.mainmenu);
    tb.setOnMenuItemClickListener(
        new Toolbar.OnMenuItemClickListener() {
            @Override
            public boolean onMenuItemClick(MenuItem item) {
                return onOptionsItemSelected(item);
            }
        });
    return true;
}

//  

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.action_settings:
            Toast.makeText(this, "Action Settings selected", Toast.LENGTH_SHORT).show();
            return true;
        default:
            break;
    }
    return super.onOptionsItemSelected(item);
}

@Override
public void onRssItemSelected(String link) {
    // more code as before
}
}

```

## تست اپلیکیشن

اپلیکیشن خود را اجرا نموده و هر دو action را انتخاب کنید. با استی با انتخاب آیتم های مختلف در نوار بازار یک پیغام Toast برای کاربر به نمایش گذاشته شود.

## تمرین: فعال سازی عملیات refresh/بروز رسانی آیتم های toolbar از طریق myListFragment

برای تست اپلیکیشن شما می خواهید recycler view تنها زمانی که کاربر refresh را از toolbar انتخاب می کند (از طریق swipe-to-refresh یا قابلیت لمس اپلیکیشن به پایین جهت بروز رسانی آیتم های لیست) با داده پر شود. بنابراین، recycler view یا لیست شما در ابتدای امر بایستی تهی باشد.

### 3-10-6- حذف داده های اولیه

در پروژه‌ی RSS Reader خود، فراخوانی updateListContent() را از بدنه‌ی متدهای onCreate() در کلاس myListFragment خود حذف نمایید.

### 3-10-7- بروز رسانی myListFragment که refresh انتخاب شد

منطق Refresh را در کلاس RssfeedActivity خود پیاده سازی نمایید. حال زمانی که کاربر refresh را انتخاب می کند، متدهای updateListContent() در myListFragment صدا خورده می شود.

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.action_refresh:
            //TODO
            // find fragment via FragmentManager and call updateListContent() on it.
            break;
        default:
            break;
    }
    return true;
}
```

بخش سوم :

## پیاده سازی الگو توسعه و قابلیت Swipe-To-Refresh (کشیدن صفحه به پایین جهت بروز رسانی لیست) در اپلیکیشن

آموزش حاضر به شرح پیاده سازی الگو توسعه‌ی swipe to refresh در اپلیکیشن می‌پردازد.



### 11-3-پیاده سازی قابلیت بروز رسانی آیتم های لیست با استفاده از swipe to refresh

اندروید همچنین یک کامپوننت UI یا widget ارائه می‌دهد که در واقع الگو توسعه‌ی swipe-to-refresh را پیاده سازی کرده و به کاربر این امکان را می‌دهد تا با کشیدن صفحه به پایین عمودی بر روی نمایشگر) آیتم‌های لیست را بروز رسانی کند (عملیات update را فراخوانی نماید).

این قابلیت توسط widget ای به نام SwipeRefreshLayout پیاده سازی می‌شود. این کامپوننت UI در حقیقت عمل کشیدن صفحه به پایین یا swipe عمودی را تشخیص داده، یک progress bar (عملیات refresh) را در اپلیکیشن فراخوانی می‌کند. مجزا به نمایش می‌گذارد و متدهای callback را در اپلیکیشن فراخوانی می‌کند.

جهت پشتیبانی از widget نام برده، ابتدا می‌بایست آن را به عنوان میزبان و پدر view مربوطه در فایل layout مورد نظر تعریف نمایید و سپس رفتار یا عملیات بروز رسانی (refresh) که با swipe عمودی کاربر بر روی نمایشگر فعال می‌شود را پیاده سازی کنید.

به منظور استفاده از swipe to refresh لازم است کتابخانه `i` مربوطه (dependency to the) را در فایل Gradle build اپلیکیشن خود اضافه کرده باشید.

```
dependencies {
    compile 'com.android.support:support-v4:24.0.0'
}
```

```
<android.support.v4.widget.SwipeRefreshLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/swiperefresh"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <android.support.v7.widget.RecyclerView
        android:id="@+id/my_recycler_view"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scrollbars="vertical" />
</android.support.v4.widget.SwipeRefreshLayout>
```

**توجه:** پس از پیاده سازی الگو توسعه `i`، لازم است برای آن دسته از کاربرانی که قابلیت لمس نمایشگر به پایین را ندارند، یک دکمه `i` refresh به overflow menu خود داخل toolbar اضافه نمایید.

داخل بدنی متدهای `SwipeRefreshLayout.OnRefreshListener` view را پیاده سازی نمایید.

```
mySwipeRefreshLayout.setOnRefreshListener(
    new SwipeRefreshLayout.OnRefreshListener() {
        @Override
        public void onRefresh() {
            doYourUpdate();
        }
    });
private void doYourUpdate() {
    // TODO implement a refresh
    setRefreshing(false); // Disables the refresh icon
}
```

چنانچه action مذبور (آیتم refresh) را در منوی overflow مستقر در toolbar خود نیز اضافه کردید، در آن صورت لازم است آیکون refresh را نیز در UI به نمایش بگذارید.

```
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.menu_refresh:
            // signal SwipeRefreshLayout to start the progress indicator
```

```
mySwipeRefreshLayout.setRefreshing(true);  
doYourUpdate();  
  
return true;  
}  
return super.onOptionsItemSelected(item);  
}
```





## بخش اول :

# دیتابیس content provider و SQLite

این مبحث به شرح مفهوم دیتابیس در اندروید می‌پردازد. سپس به ترتیب نحوه‌ی استفاده از دیتابیس SQLite در اپلیکیشن، استفاده از ContentProvider جاری و ایجاد نمونه‌های جدید از آن را برای شما تشریح می‌کند. در نهایت نمونه‌ای از کاربرد چارچوب نرم افزاری (framework) Loader جهت بارگذاری داده به صورت ناهمzman (asynch) در برنامه‌ی اندرویدی را برای شما به نمایش می‌گذارد.

پروژه‌ی این مبحث در محیط کاری Eclipse 4.2 نوشته شده، مبتنی بر ویرایش 1.6 زبان Java و ورژن 4.2 سیستم عامل اندروید می‌باشد.

## Android و SQLite-1-4

### SQLite-4-1-1 چیست و چه کاربردی دارد؟

یک دیتابیس کد باز (Open Source) بوده، از امکانات و ویژگی‌های متعارف دیتابیس های رابطی نظیر ساختار نگارشی (SQL syntax)، تراکنش‌ها (transactions) و دستورات آماده پشتیبانی می‌کند. دیتابیس مزبور در زمان اجرای اپلیکیشن حافظه‌ی کمی را مصرف می‌کند (حدودا 250 کیلویایت). این امر سبب شده که SQLite گزینه‌ی خوبی برای جاسازی در سایر runtime ها تلقی شود.

SQLlite از انواع داده‌ای همچون TEXT (معادل String در جاوا)، INTEGER (معادل long در جاوا) و REAL (مشابه double در جاوا) پشتیبانی می‌کند. سایر نوع‌های داده‌ای می‌باشد قبل از قابلیت ذخیره در دیتابیس، به یکی از فیلد‌های نام برده تبدیل شوند.

SQLlite به صورت خودکار قابلیت اعتبارسنجی انواع داده‌ای درج شده در ستون‌ها را ندارد. به عنوان مثال، ممکن است مقداری از نوع integer را در ستون رشته‌ای ذخیره کرده یا بالعکس. خواهید دید که دیتابیس مذکور از شما ایراد نخواهد گرفت.

## SQLLite در اندروید 4-1-2

تقریباً در تمامی دستگاه های اندروید جاسازی شده است. برای استفاده از این دیتابیس، توسعه دهنده نیازی به نصب و تنظیم آن ندارد. مدیریت (admin) دیتابیس نیز به صورت خودکار توسط خود اندروید انجام می شود.

کافی است دستورات SQL ایجاد و آپدیت دیتابیس را تعریف نموده و فراخوانی کنید. پس از آن باقی عملیات لازم را خود محیط و بستر اجرای اندروید (platform) به دست می گیرد.

دسترسی به دیتابیس SQLite به صورت خودکار دسترسی به سیستم فایل را نیز به دنبال دارد و به همین دلیل عملیات مرتبط با دیتابیس می تواند کند باشد.

دیتابیس اپلیکیشن به صورت خودکار در آدرس DATA/data/APP\_NAME/databases/FILENAME ذخیره می شود.

بخش های آدرس فوق بر اساس قوانین و قواعد خاصی ساخته می شوند. DATA آدرس یا مسیری است که متدهای Environment.getDataDirectory() APP\_NAME اسم اپلیکیشن شما است و در پایان، FILENAME اسمی است که در کد اپلیکیشن خود برای دیتابیس تعریف می کنید.

## SQLLite 4-2-1 معماری Package ها

پکیج android.database تمامی اطلاعات ضروری برای کار با دیتابیس را شامل می شود. پکیج android.database.sqlite تمامی کلاس های ویژه SQLite را دربرمی گیرد.

## 2-2-4- ایجاد و آپدیت دیتابیس با SQLiteOpenHelper

به منظور ایجاد و ارتقا دیتابیس در اپلیکیشن اندرویدی خود، لازم است یک کلاس فرزند (subclass) از SQLiteOpenHelper ایجاد نمایید. سپس در تابع سازنده (constructor) کلاس فرزند، متدهای onCreate() و onUpgrade() را فراخوانی نموده و اسم و ورژن دیتابیس را مشخص نمایید.

در این کلاس می‌بایست پیاده سازی متدهای زیر را جهت ایجاد و آپدیت دیتابیس خود، بازنویسی (override) نمایید.

- می‌خواهید به دیتابیس دسترسی داشته باشید، اما دیتابیس هنوز ایجاد نشده است.
- هنگامی فراخوانی می‌شود که ورژن دیتابیس در کد اپلیکیشن ارتقا یافته باشد. این متدها شما اجازه می‌دهد تا schema دیتابیس جاری را بروز رسانی نموده یا دیتابیس جاری را حذف کنید و سپس آن را مجدداً با متدهای onCreate() و onUpgrade() ایجاد نمایید.

هر دو متدهای آبجکت SQLiteDatabase به عنوان پارامتر می‌گیرند. این آبجکت‌ها در واقع همان دیتابیس هستند.

کلاس SQLiteOpenHelper با ارائه دو متدهای getWritableDatabase() و getReadableDatabase() دسترسی به آبجکت SQLiteDatabase را به صورت فقط خواندنی و نوشتنی برای توسعه دهنده فراهم می‌آورد.

جداول دیتابیس باستفاده از id را به عنوان کلید اصلی (primary key) می‌گنجانند. بسیاری از رفتارها و توابع اندروید از این استاندارد پیروی می‌کنند.

**نکته:** توصیه می‌شود برای هر کلاس یک جدول جداگانه ایجاد نمایید. این کلاس‌ها متدهای استاتیکی () و onCreate() و onUpgrade() را تعریف می‌کنند. توابع مذکور در متدهای متناظر داخل SQLiteOpenHelper فراخوانی می‌شوند. با پیروی از این روش، حتی اگر تعداد جداول دیتابیس بسیار بالا باشد، پیاده سازی SQLiteOpenHelper به راحتی قابل دسترسی و خواندن خواهد بود.

## 3-2-4- کلاس پدر SQLiteDatabase

کلاس SQLiteDatabase پایه ای است که برای کار با دیتابیس می باشد از آن ارث بری کنید. این کلاس متدهایی را جهت بازگردان، کوئری گرفتن، آپدیت کردن و بستن دیتابیس در اختیار توسعه دهنده قرار می دهد. به عبارت دقیق تر، SQLiteDatabase سه متد بسیار مهم insert() و update() و delete() را ارائه می دهد.

علاوه بر آن، کلاس نام بردۀ با ارائه ی متد execSQL() این امکان را برای برنامه نویس فراهم کرده تا دستور SQL را به صورت مستقیم اجرا کند.

آبجکت ContentValues جفت های کلید/مقدار (key/value) را ایجاد می کند. Key همان id یا شناسه ی ستون جدول است و value نیز محتوا یا مقدار رکورد جدول در این ستون می باشد. از این آبجکت برای درج مقدار جدید و آپدیت مقادیر جاری دیتابیس استفاده می شود.

جهت ایجاد query ها و درخواست داده از دیتابیس می توانید به دو روش اقدام کنید: 1. می توانید یکی از دو متد query() و rawQuery() را فراخوانی نمایید. 2. از کلاس SQLiteQueryBuilder استفاده کنید.

یک دستور SQL را مستقیماً به عنوان ورودی می پذیرد.  
یک interface query() ساخت یافته و الگوی پیاده سازی آماده برای تعریف دستور SQL ارائه می دهد.

یک کلاس کمکی (Convenience) است که این امکان را به شما می دهد تا query های sql را به راحتی بسازید.

### مثالی از پیاده سازی متد rawQuery()

نمونه کد زیر متد rawQuery() را فراخوانی می کند.

```
Cursor cursor = getReadableDatabase();
    rawQuery("select * from todo where _id = ?", new String[] { id });
```

### مثالی از پیاده سازی متد query()

نمونه کد زیر متد () query را فراخوانی می کند.

```
return database.query(DATABASE_TABLE,
    new String[] { KEY_ROWID, KEY_CATEGORY, KEY_SUMMARY, KEY_DESCRIPTION },
    null, null, null, null);
```

متد () query پارامترهای زیر را به عنوان ورودی می گیرد.

جدول پارامترهای ورودی متد () query	شرح
پارامتر	
String dbName	اسم جدولی که از آن کوئری گرفته می شود.
String[] columnNames	لیستی از ستون ها که باید در خروجی بازگردانده شوند. با ارسال مقدار "null" به عنوان پارامتر ورودی، تمامی ستون ها در خروجی بازگردانی می شوند.
String whereClause	عبارت شرطی Where که به مثابه ی فیلتری جهت واکشی داده های مورد نیاز استفاده می شود. ارسال مقدار null، تمامی داده ها را در خروجی برمی گرداند.
String[] selectionArgs	می توانید ? را در عبارت ""whereClause"" نیز بکار ببرید. این مکان نگهدار (placeholder) سپس با مقادیر واقعی از آرایه ی selectionArgs جایگزین می شوند.
String[] groupBy	یک فیلتر است که نحوه ی گروه بندی سطرها را مشخص می کند. ارسال مقدار null به عنوان پارامتر سبب می شود که سطرها مرتب نشوند.
String[] having	این پارامتر نتیجه را به آن گروه از سطرها که با شرط اعمال شده منطبق هستند، محدود می کند. به عبارت دیگر برای اعمال شرط و محدود کردن خروجی در گروه ها مورد استفاده قرار می گیرد.
String[] orderBy	ستون هایی از جدول که برای مرتب سازی داده ها مورد استفاده قرار می گیرد را شامل می شود. با ارسال null به عنوان پارامتر هیچ مرتب سازی صورت نمی گیرد.

اگر شرطی برای اعمال و محدودسازی نتایج نیاز نباشد، می توانید `null` را به عنوان آرگومان ارسال نمایید. به عنوان مثال چنانچه لزومی برای اعمال شرط بر روی گروه ها نیست، می توانید `null` را پاس دهید.

در متدهای `query()` همان طور که مشاهده می کنید، بجای عبارت شرطی "whereClause" از این ساختار استفاده شده است: "`_id=19 and summary=?`". در واقع عملگر "=" جایگزین عبارت `where` می شود.

اگر با استفاده از مقادیر موقتی و مکان نگهدار تعريف نمایید (`placeholder`)، در آن صورت می باشد آن ها را به عنوان `selectionArgs` به کوئری ارسال کنید.

## 4-2-4-آبجکت Cursor

کوئری در خروجی یک آبجکت `Cursor` برمی گرداند. `Cursor` همان نتیجه ی کوئری و سطرهای واکشی شده از دیتابیس است و اساساً به یک سطر از نتیجه ی کوئری اشاره دارد. این روش بازیابی اطلاعات به اندروید اجازه می دهد تا نتایج کوئری را به راحتی `buffer` کند چرا که در این صورت سیستم دیگری نیازی به بارگذاری تمامی داده ها در حافظه ندارد.

برای بدست آوردن تعداد المان های کوئری خروجی می توانید متدهای `getCount()` را بکار ببرید.

به منظور حرکت بین سطرهای فردی، می توانید `moveToFirst()` و `moveToNext()` را فراخوانی نمایید. با استفاده از متدهای `isLast()` و `isFirst()` می توانید برسی کنید آیا به انتهای نتیجه ی کوئری رسیده اید یا خیر.

مجموعه متدهای `typed` (متدهای `get*`) وابسته به نوع (`String`, `Long`, `Boolean`, `Double`, `Float`) را در اختیار توسعه دهنده قرار می دهد. از جمله می توان به `getString(columnIndex)`, `getLong(columnIndex)`, `getBoolean(columnIndex)`, `getFloat(columnIndex)` و `getDouble(columnIndex)` اشاره کرد. برای بازیابی داده های ستون با توجه به اندیس کنونی نتیجه اشاره کرد. پارامتر `columnIndex` همان طور که از نامش پیدا است، شماره ی مکان استفاده می شود. این پارامتر می تواند از نوع `int` باشد که داده را بر اساس آن استخراج می کند.

همچنین متدهای `getColumnIndexOrThrow(String)` را ارائه می‌دهد. این متدهای اسم ستون را به عنوان ورودی گرفته و در خروجی شماره‌ی مکان قرار گیری ستون را بر می‌گردانند. لازم به ذکر است که آبجکت `Cursor` را می‌بایست با فراخوانی متدهای `close()` ببندید.

## DADEHA

### SimpleCursorAdapter و ListView ها، ListActivity 4-2-5

`ListView` ها آبجکت‌های `view` ای هستند که لیستی از المان‌ها را برابر به نمایش می‌گذارند.

برای `activity` های ویژه‌ای هستند که استفاده از `ListView` ها را آسان‌تر می‌سازند.

برای کار با دیتابیس‌ها و `ListView` ها می‌توانید از `SimpleCursorAdapter` استفاده نمایید. به شما این امکان را می‌دهد تا برای هر سطر از `ListView` ها یک `layout` تنظیم نمایید.

شما بایستی آرایه‌ای تعریف کنید که در بردارنده‌ی اسم ستون‌ها می‌باشد و سپس آرایه‌ی دیگری جهت نگهداری `ID` و شناسه‌ی `View` هایی که باید با داده‌پر شوند ایجاد نمایید. کلاس `SimpleCursorAdapter` ستون‌ها را بر اساس `Cursor` ارسالی، به `view` ها نگاشت می‌کند.

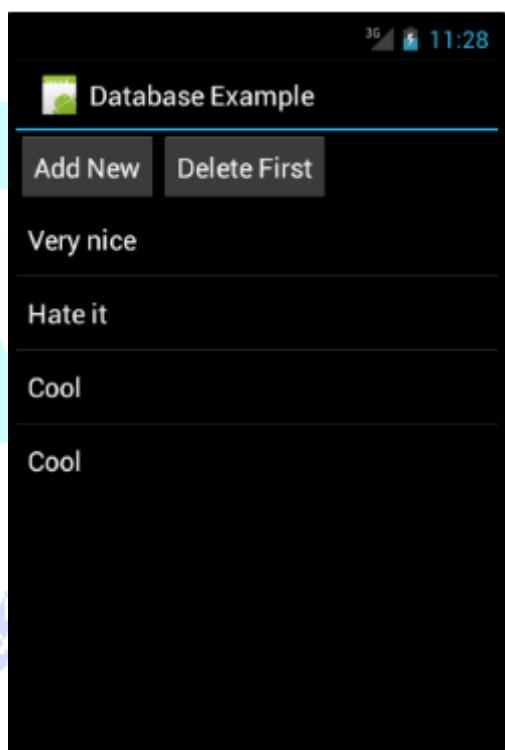
جهت دسترسی به `Cursor` می‌بایست کلاس `Loader` را بکار ببرید.

### SQLite آموزش: استفاده از 4-3-

### 4-3-1-مقدمه ای بر پروژه

در این بخش نحوه‌ی کار با دیتابیس SQLite را به صورت کاربردی خواهید آموخت. برای مدیریت داده‌ها از یک DAO (آبجکت دسترسی به داده‌ها) استفاده خواهید نمود. DAO وظیفه‌ی مدیریت اتصال به دیتابیس، دسترسی و ویرایش داده‌ها را بر عهده دارد. طی این مکانیزم همچنین آبجکت‌های دیتابیس به آبجکت‌های متعارف جاوا (Java Objects) تبدیل می‌شوند و از این طریق کد UI دیگر با لایه‌ی ذخیره‌ی دائمی داده‌ها یا persistence layer تعامل نخواهد داشت.

در پایان اپلیکیشن به صورت زیر خواهد بود:



گفتنی است که استفاده از الگو DAO همیشه و در همه‌ی شرایط بهترین روش نیست. در واقع این الگو، model object‌های جاوا (آبجکت‌هایی از روی جداول دیتابیس) ایجاد می‌کند و منابع بیشتری مصرف می‌کند. استفاده‌ی مستقیم از یک دیتابیس یا دسترسی به داده‌ها از طریق یک ContentProvider معمولاً بهینه‌تر بوده و از آنجایی که در این دو روش دیگر نیازی به ساخت آبجکت‌های جاوا نیست، میزان حافظه مورد استفاده کاهش می‌یابد.

همان طور که گفته شد در این روش از DAO استفاده می شود. برای این مثال ورژن کتابخانه های اندروید را بر روی API 15 تنظیم نمایید که معادل اندروید 4.0 می باشد. در غیر این صورت می بایست از کلاس Loader برای مدیریت Cursor استفاده می شد که از اندروید 3.0 به بعد پشتیبانی می شود. بعلاوه استفاده از این کلاس پیچیدگی خاص خود را دارد.

## ایجاد پروژه

یک پروژه‌ی جدید اندروید و یک activity به نام‌های activity و de.vogella.android.sqlite.first و TestDatabaseActivity ایجاد نمایید.

### 4-3-2 ساخت Data Model و Database

ابتدا کلاس MySQLHelper را ایجاد نمایید. این کلاس دیتابیس اپلیکیشن را ایجاد می‌کند.

اگر به خاطر داشته باشید، متدهای onUpgrade() به کلی داده‌های فعلی بانک اطلاعاتی را حذف کرده و جدول را مجدداً ایجاد می‌کند. سپس تعدادی ثابت (constant) جهت نگهداری مقادیر اسم جدول و ستون‌های جدول ایجاد می‌نمایید.

```
package de.vogella.android.sqlite.first;
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;
public class MySQLHelper extends SQLiteOpenHelper {
    public static final String TABLE_COMMENTS = "comments";
    public static final String COLUMN_ID = "_id";
    public static final String COLUMN_COMMENT = "comment";
    private static final String DATABASE_NAME = "commments.db";
    private static final int DATABASE_VERSION = 1;
    // Database creation sql statement
    private static final String DATABASE_CREATE = "create table "
            + TABLE_COMMENTS + "(" + COLUMN_ID
            + " integer primary key autoincrement, " + COLUMN_COMMENT
            + " text not null);";
    public MySQLHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }
    @Override
    public void onCreate(SQLiteDatabase database) {
        database.execSQL(DATABASE_CREATE);
    }
    @Override
```

```

public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    Log.w(MySQLiteHelper.class.getName(),
        "Upgrading database from version " + oldVersion + " to "
        + newVersion + ", which will destroy all old data");
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_COMMENTS);
    onCreate(db);
}
}

```

کلاس Comment را ایجاد نمایید. این کلاس به منزله ی model (آبجکتی ساخته شده از جدول دیتابیس) خواهد بود و تمامی داده هایی که در دیتابیس ذخیره شده و برای کاربر به نمایش در می آید را دربرخواهد گرفت.

```

package de.vogella.android.sqlite.first;
public class Comment {
    private long id;
    private String comment;
    public long getId() {
        return id;
    }
    public void setId(long id) {
        this.id = id;
    }
    public String getComment() {
        return comment;
    }
    public void setComment(String comment) {
        this.comment = comment;
    }
    // Will be used by the ArrayAdapter in the ListView
    @Override
    public String toString() {
        return comment;
    }
}

```

کلاس CommentDataSource را ایجاد نمایید. این کلاس همان آبجکتی است که برای دسترسی به دیتابیس (DAO) از آن استفاده خواهد نمود. آبجکت ذکر شده اتصال به دیتابیس را نگه داشته و امکان درج comment های جدید و واکشی تمامی comment های جاری را برای اپلیکیشن فراهم می کند.

```

package de.vogella.android.sqlite.first;
import java.util.ArrayList;
import java.util.List;
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;

```

```

import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
public class CommentsDataSource {
    // Database fields
    private SQLiteDatabase database;
    private MySQLiteHelper dbHelper;
    private String[] allColumns = { MySQLiteHelper.COLUMN_ID,
        MySQLiteHelper.COLUMN_COMMENT };
    public CommentsDataSource(Context context) {
        dbHelper = new MySQLiteHelper(context);
    }
    public void open() throws SQLException {
        database = dbHelper.getWritableDatabase();
    }
    public void close() {
        dbHelper.close();
    }
    public Comment createComment(String comment) {
        ContentValues values = new ContentValues();
        values.put(MySQLiteHelper.COLUMN_COMMENT, comment);
        long insertId = database.insert(MySQLiteHelper.TABLE_COMMENTS, null,
            values);
        Cursor cursor = database.query(MySQLiteHelper.TABLE_COMMENTS,
            allColumns, MySQLiteHelper.COLUMN_ID + " = " + insertId, null,
            null, null, null);
        cursor.moveToFirst();
        Comment newComment = cursorToComment(cursor);
        cursor.close();
        return newComment;
    }
    public void deleteComment(Comment comment) {
        long id = comment.getId();
        System.out.println("Comment deleted with id: " + id);
        database.delete(MySQLiteHelper.TABLE_COMMENTS, MySQLiteHelper.COLUMN_ID
            + " = " + id, null);
    }
    public List<Comment> getAllComments() {
        List<Comment> comments = new ArrayList<Comment>();
        Cursor cursor = database.query(MySQLiteHelper.TABLE_COMMENTS,
            allColumns, null, null, null, null, null);
        cursor.moveToFirst();
        while (!cursor.isAfterLast()) {
            Comment comment = cursorToComment(cursor);
            comments.add(comment);
            cursor.moveToNext();
        }
        // make sure to close the cursor
        cursor.close();
        return comments;
    }
    private Comment cursorToComment(Cursor cursor) {

```

```

        Comment comment = new Comment();
        comment.setId(cursor.getLong(0));
        comment.setComment(cursor.getString(1));
        return comment;
    }
}

```

### 3-3-4- ساخت ظاهر و UI اپلیکیشن

فایل main.xml در `<filename class="directory">res/layout_ folder` را به صورت زیر ویرایش نمایید. این layout دو دکمه برای افزودن/حذف comment ها و یک ListView (لیست ساده) جهت نمایش comment های جاری تعریف می کند. متن بعدها در کلاس activity و به وسیله ی یک random generator تولید خواهد شد.

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <LinearLayout
        android:id="@+id/group"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" >
        <Button
            android:id="@+id/add"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Add New"
            android:onClick="onClick"/>
        <Button
            android:id="@+id/delete"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Delete First"
            android:onClick="onClick"/>
    </LinearLayout>
    <ListView
        android:id="@+id/list"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
</LinearLayout>

```

کلاس TestDatabaseActivity را به صورت زیر ویرایش نمایید. برای نمایش آسان لیست ساده ای از اطلاعات می توانید از یک ListActivity استفاده نمایید.

```

package de.vogella.android.sqlite.first;
import java.util.List;
import java.util.Random;
import android.app.ListActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
public class TestDatabaseActivity extends ListActivity {
    private CommentsDataSource datasource;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        datasource = new CommentsDataSource(this);
        datasource.open();
        List<Comment> values = datasource.getAllComments();
        // use the SimpleCursorAdapter to show the
        // elements in a ListView
        ArrayAdapter<Comment> adapter = new ArrayAdapter<Comment>(this,
        android.R.layout.simple_list_item_1, values);
        setListAdapter(adapter);
    }
    // Will be called via the onClick attribute
    // of the buttons in main.xml
    public void onClick(View view) {
        @SuppressWarnings("unchecked")
        ArrayAdapter<Comment> adapter = (ArrayAdapter<Comment>) getListAdapter();
        Comment comment = null;
        switch (view.getId()) {
            case R.id.add:
                String[] comments = new String[] { "Cool", "Very nice", "Hate it" };
                int nextInt = new Random().nextInt(3);
                // save the new comment to the database
                comment = datasource.createComment(comments[nextInt]);
                adapter.add(comment);
                break;
            case R.id.delete:
                if (getListAdapter().getCount() > 0) {
                    comment = (Comment) getListAdapter().getItem(0);
                    datasource.deleteComment(comment);
                    adapter.remove(comment);
                }
                break;
        }
        adapter.notifyDataSetChanged();
    }
    @Override
    protected void onResume() {
        datasource.open();
        super.onResume();
    }
    @Override

```

```

protected void onPause() {
    datasource.close();
    super.onPause();
}
}

```

### 4-3-4-نصب و اجرای اپلیکیشن

اپلیکیشن خود را نصب نموده، دکمه های Add و Delete را کلیک نمایید. سپس اپلیکیشن را restart کرده و اطمینان حاصل نمایید که داده ها پس از راه اندازی مجدد همچنان در جای خود هستند.

## 4-4-شرح مفهوم Content Provider

### 4-4-1-Content Provider چیست و چه کاربردی دارد؟

جهت به اشتراک گذاری داده ها بین چندین اپلیکیشن، می توانید از Content Provider استفاده نمایید. می توانید به Provider به چشم یک منبع داده ای نگاه کنید که اطلاعات موجود در آن به اپلیکیشنی که از آن استفاده می کند، وابسته نمی باشد. این امکان وجود دارد که از Provider مانند یک دیتابیس کوئری بگیرید، داده های آن را بروز رسانی و حتی حذف نمایید. علاوه بر آن provider ها قابلیتی دارند که به آن ها اجازه ذخیره داده ها در فایل، دیتابیس و شبکه را می دهد.

اطلاعات را بر اساس URI مربوطه کپسوله سازی می کند. هر URI ای که با content://Provider شروع می شود در واقع به منابع و محتوایی اشاره دارد که از طریق Provider قابل دسترسی است.

آدرس URI یک منبع به شما این امکان را می دهد تا از طریق Provider عملیات ساده‌ی CRUD (ایجاد، بازیابی، بروز رسانی و حذف) را بر روی داده های منبع مورد نظر اجرا نمایید.

همان طور که قبلا اشاره شد، provider داده های مورد نیاز اپلیکیشن را در اختیار آن قرار می دهد. این داده ها ممکن است در یک دیتابیس SQLite، در سیستم فایل، داخل فایل های flat یا بر روی سرور راه دور (remote server) مستقر باشند.

اگرچه از Provider می توان برای دسترسی به داده ها در یک اپلیکیشن نیز استفاده کرد، اما معمولاً توسعه دهنگان از آن برای به اشتراک گذاری داده ها بین چندین اپلیکیشن بهره می گیرند. داده های اپلیکیشن به صورت پیش فرض دارای سطح دسترسی private می باشند. اینجا است که content provider به یاری شما می آید؛ این کامپوننت نرم افزاری دسترسی به داده های سایر اپلیکیشن ها را بر اساس یک interface ساخت یافته به راحتی مهیا می سازد.

Content provider (manifest) را می بایست داخل فایل تنظیمات اپلیکیشن اندرویدی خود تعریف نمایید.

#### 4-4-2 قالب پایه ای و الگوی تعریف URI جهت دسترسی به Provider

برای دسترسی به content provider می بایست ابتدا پیشوند متعارف content:// و سپس content provider مربوط به provider را ارائه نمایید (namespace همان نام provider). لازم است namespace را داخل فایل manifest تگ .content:// داشته باشد. مثلاً .content://test/

در زیر بخش ها مختلف یک URI را شرح می دهیم.

قالب `<standard_prefix>://<authority>/<data_path>/<id>` و ساختار کلی URI می باشد:

1. `.content://` یا پیشوندی که همیشه ثابت است.

2. `.content provider` یا نام authority.

3. `data path` نوع درخواست را مشخص می کند.

4. `id` رکورد مورد نظر را مشخص می کند.

پایه امکان دسترسی به مجموعه ای از منابع (برای مثال کل یک جدول) را فراهم می کند. چنانچه در بخش نهايی URI شناسه‌ی نمونه ذکر شود، در آن صورت تنها content ای که id آن ذکر شده، قابل دسترسی می باشد. مثال: `content://test/2`

#### 4-4-3- دسترسی به content provider

از آنجایی که داشتن URI های یک provider برای دسترسی به آن ضروری است، توصیه می شود URI ها را داخل constant هایی با سطح دسترسی public قرار داده، آن ها را به صورت مستند در اختیار دیگر توسعه دهنگان قرار دهید.

بسیاری از منابع داده ای اندروید همچون contacts (اطلاعات مخاطبین) از طریق content provider قابل دسترسی هستند.

#### 4-4-4- پیاده سازی content provider اختصاصی

به منظور ایجاد content provider اختصاصی خود بایستی یک کلاس که از کلاس `android.content.ContentProvider` ارث می برد (`extend` می کند)، ایجاد نمایید. سپس این مربوطه در فایل تنظیمات (Android manifest) معرفی کنید. تگ `content provider` را به عنوان `Authority` در واقع مبنای URI را تشکیل داده و برای دسترسی به content provider مورد استفاده قرار می گیرد، از اینرو باید منحصر بفرد باشد.

```
<provider android:authorities="de.vogella.android.todos.contentprovider"
          android:name=".contentprovider.MyTodoContentProvider" >
</provider>
```

در بدنه ای Content provider شما باید تعداد زیادی متد را پیاده سازی کنید. برای مثال می توان به ()، onCreate()، getType()، delete()، update()، insert()، query() اشاره کرد. در صورتی که از متد خاصی

پشتیبانی نمی کنید، توصیه می شود ()UnsupportedOperationException را فراخوانی نمایید.

متد ()query در خروجی یک آبجکت Cursor برمی گردداند.

## و مبحث امنیت Content provider\_4-4-5

تا ویرایش 4.2 اندروید، content provider به صورت پیش فرض برای دیگر اپلیکیشن های اندرویدی قابل دسترسی می باشد. اما از ورژن 4.2 به بعد اندروید، برای دسترسی به content provider لازم است آن را به صورت صریح export نمایید.

به منظور تنظیم قابلیت رویت و دسترسی content provider خود، کافی است پارامتر android:exported=false|true ذکر نمایید. AndroidManifest.xml

**نکته:** توصیه می شود پارامتر android:exported را همیشه در فایل تنظیمات قید کنید تا رفتار اپلیکیشن در ورژن های مختلف اندروید یکسان باشد.

## Thread safety-4-4-6 (brطرف سازی مشکل همزمانی با استفاده از synchronized کلیدوازه‌ی)

چنانچه مستقیماً با دیتابیس تعامل داشته باشید و همزمان از چند thread در دیتابیس مقداری نوشته شوند (از چند thread همزمان چندین writer وجود داشته باشد) در آن صورت طبیعتاً با مشکل مدیریت همرونگی و concurrency مواجه می‌شوید.

یک content provider می‌تواند همزمان از چندین برنامه مورد دسترسی قرار گیرد. به این دلیل لازم است دسترسی thread-safe را پیاده سازی نمایید. آسان ترین راه، استفاده از کلیدوازه‌ی synchronized در مقابل تمامی متدهای provider است. بدین وسیله تنها یک thread در لحظه می‌تواند به این متدها دسترسی داشته باشد.

= دسترسی چند thread به یک data structure بدون اینکه در پروسه‌ی کاری thread‌ها اختلالی به وجود آید.

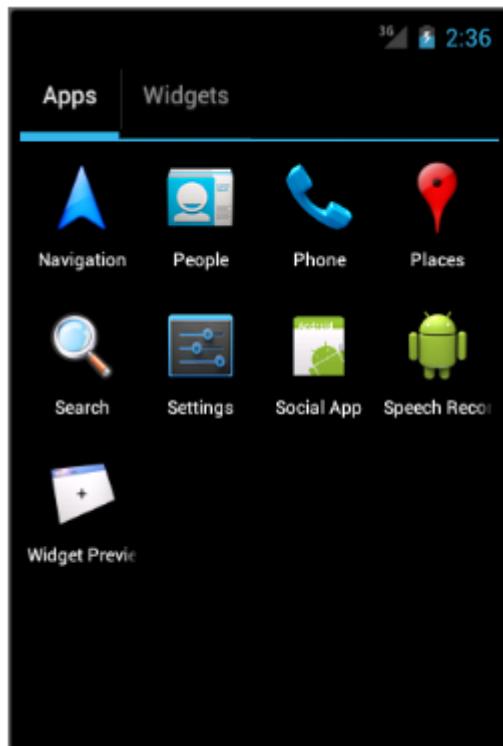
اگر لزومی ندارد که اندروید دسترسی به provider را sync کند، مقدار خصیصه (attribute) android:multiprocess=true را در تگ <provider>، داخل فایل تنظیمات اپلیکیشن خود برابر قرار دهید. این کار سبب می‌شود در هر یک از process‌های مربوط به کلاینت (پاسخ دهنده به درخواست‌های کاربر)، یک نمونه از provider ایجاد شده و نیاز به اجرای IPC به کلی از میان برداشته شود (IPC = عبارت است از مجموعه‌ای از روش‌ها برای تبادل داده بین چندین پردازش خرد و کوچک/thread). بنابراین هر یک از این روش‌ها می‌تواند برای ارتباط بین یک یا چند پردازش بکار رود.

## آموزش: استفاده‌ی کاربردی از ContentProvider

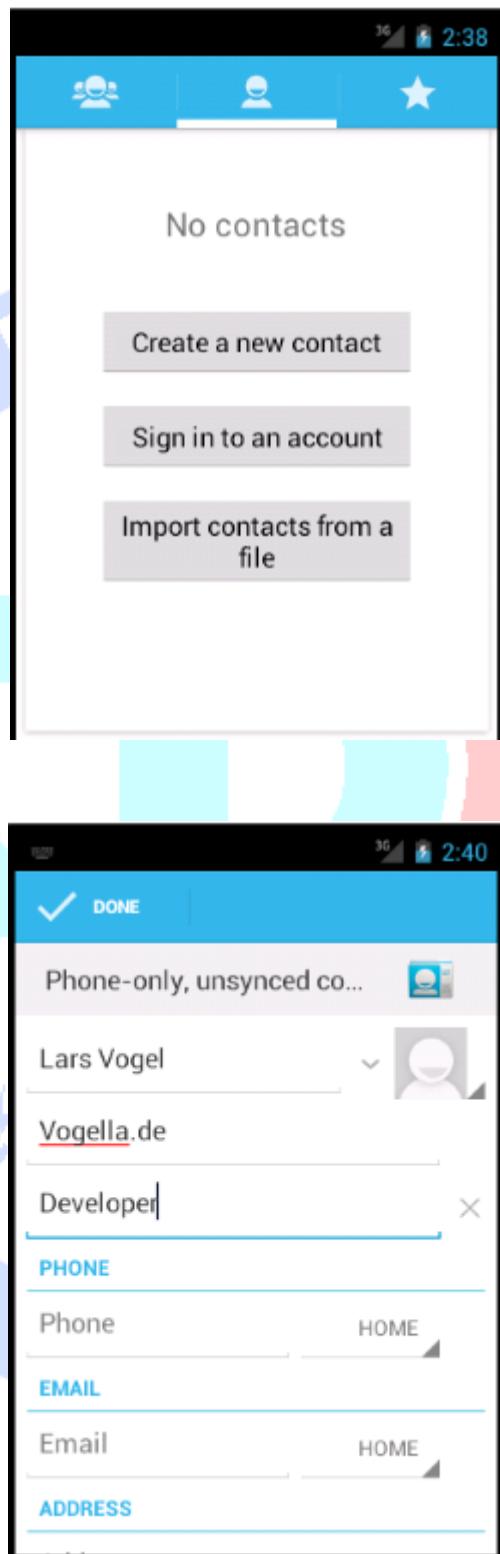
مثال زیر از آماده‌ی اپلیکیشن People content provider استفاده می‌کند.

#### 4-4-6- ایجاد contact ها (مخاطبین) در محیط شبیه ساز

برای این مثال لازم است تعدادی contact ثابت داشته باشید. منوی home را انتخاب نموده و سپس المان People را برای ایجاد contacts انتخاب کنید.



اپلیکیشن از شما می پرسد آیا میلی به ورود و login در برنامه دارید یا خیر. می توانید login کنید یا "Not now" را انتخاب نمایید. حال بر روی دکمه "Create a new contact" کلیک نمایید. می توانید local contacts را ایجاد کنید.



360

آدرس آموزشگاه : تهران - خیابان شریعتی - بالاتر از خیابان ملک - جنب بانک صادرات - پلاک 651 طبقه دوم - واحد 7

88146323 - 88446780 - 88146330

پس از اینکه اولین contact را ایجاد کردید، اپلیکیشن به شما اجازه می دهد تا با کلیک بر روی دکمه `+` های بیشتری ایجاد کنید. در پایان چندین contact به اپلیکیشن خود اضافه کرده اید.

## 4-4-استفاده از Contact Content Provider

یک پروژه و activity جدید اندروید به ترتیب به نام های `ContactsActivity` و `Contacts` ایجاد نمایید.

فایل layout مربوطه را نیز در آدرس `res/layout_folder` ویرایش نمایید. ID المان `contactview` را به `TextView` تغییر داده و متن پیش فرض را حذف نمایید.

فایل layout خروجی می بایست دارای ظاهری مشابه زیر باشد.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <TextView
        android:id="@+id/contactview"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</LinearLayout>
```

از آنجایی که تمامی اپلیکیشن ها نباید به اطلاعات شخصی مخاطبین دسترسی داشته باشند، لازم است برای دستیابی به `contact ContentProvider`، در فایل تنظیمات `permission` خاصی تنظیم نمایید. برای این منظور ابتدا فایل `AndroidManifest.xml` را باز نموده و تب `Permissions` را انتخاب کنید. در این تب، بر روی دکمه `Add` کلیک کرده و `Uses Permission` را انتخاب نمایید. از لیست کشویی (drop-down) حاضر، آیتم `android.permission.READ_CONTACTS` را انتخاب کنید.

پیاده سازی activity را به صورت زیر ویرایش نمایید.

```
package de.vogella.android.contentprovider;
import android.app.Activity;
import android.database.Cursor;
```

```

import android.net.Uri;
import android.os.Bundle;
import android.provider.ContactsContract;
import android.widget.TextView;

public class ContactsActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_contacts);
        TextView contactView = (TextView) findViewById(R.id.contactview);
        Cursor cursor = getContacts();
        while (cursor.moveToNext()) {
            String displayName = cursor.getString(cursor
                    .getColumnIndex(ContactsContract.Data.DISPLAY_NAME));
            contactView.append("Name: ");
            contactView.append(displayName);
            contactView.append("\n");
        }
    }
    private Cursor getContacts() {
        // Run query
        Uri uri = ContactsContract.Contacts.CONTENT_URI;
        String[] projection = new String[] { ContactsContract.Contacts._ID,
                ContactsContract.Contacts.DISPLAY_NAME };
        String selection = ContactsContract.Contacts.IN_VISIBLE_GROUP + " = "
                + ("1") + "";
        String[] selectionArgs = null;
        String sortOrder = ContactsContract.Contacts.DISPLAY_NAME
                + " COLLATE LOCALIZED ASC";
        return managedQuery(uri, projection, selection, selectionArgs,
                sortOrder);
    }
}

```

زمانی که اپلیکیشن را اجرا کنید، داده های لازم از کامپوننت نرم افزاری ContentProvider که اطلاعات اپلیکیشن People را حمل می کند، بارگذاری شده و نهایتا در المان رابط کاربری TextView به نمایش گذاشته می شود. جهت ارائه ی چنین اطلاعاتی به کاربر، توسعه دهندگان معمولا از لیست ساده که توسط ListView پیاده سازی می شود، استفاده می کنند.

Unresolved directive in 001\_article.adoc - include::../AndroidBackgroundProcessing/content/120\_loader.adoc[]

## Loader ها و Cursor-5-4

یکی از مشکلات و چالش های دسترسی به دیتابیس و خواندن اطلاعات، پایین بودن سرعت آن است. بعلاوه، اپلیکیشن می بایست life cycle کامپوننت ها را در نظر گرفته و آن را مدیریت کند. برای مثال می توان به باز کردن و بستن cursor در صورت تغییر در تنظیمات و نحوه ی پیکربندی اشاره کرد.

تا ویرایش 3.0 سیستم عامل اندروید، توسعه دهندگان با فراخوانی متدهای managedQuery() در کلاس های activity، چرخه ی حیات یا lifecycle کامپوننت ها را اداره می کردند.

اما از ویرایش 3.0 به بعد اندروید، استفاده از این متدهای دیگر توصیه نمی شود و توسعه دهندگان برای دسترسی به ContentProvider می بایست از چارچوب نرم افزاری Loader (framework) استفاده کند.

کلاس SimpleCursorAdapter که همراه با ListViews نیز قابل استفاده می باشد، متدهای swapCursor() را در اختیار شما قرار می دهد. کلاس Loader اپلیکیشن شما می تواند با فراخوانی این متدهای Cursor را در متدهای onLoadFinished() بروز رسانی کند.

کلاس CursorLoader، آبجکت Cursor را پس از تغییر در تنظیمات و نحوه ی پیکربندی مجدداً متصل می کند.

# آموزش: SQLite، Loader و ContentProvider با پیاده سازی

اختصاصی

مرور کلی

نمونه اپلیکیشنی که در زیر مشاهده می کنید، در Android Market نیز به صورت آماده در دسترس است. برای اینکه کاربران بیشتری بتوانند از آن استفاده کنند، اپلیکیشن به ویرایش 2.3 سیستم عامل اندروید downport شده و بر روی آن ها قابل اجرا می باشد. در صورتی که اپلیکیشن بارکد خوان را بر روی دستگاه خود نصب کرده باشید، می توانید با اسکن کد QR زیر به سرعت اپلیکیشن مورد نظر را در Android Market بیدا کنید. لازم به توضیح است که اپلیکیشن نام بردۀ در ورزن های مختلف اندروید ممکن است ظاهر و رفتار متفاوتی داشته باشد. برای مثال ممکن است بجای OptionMenu در اپلیکیشن ActionBar را داشته باشید و پوسته (theme) برنامه کمی متفاوت باشد.

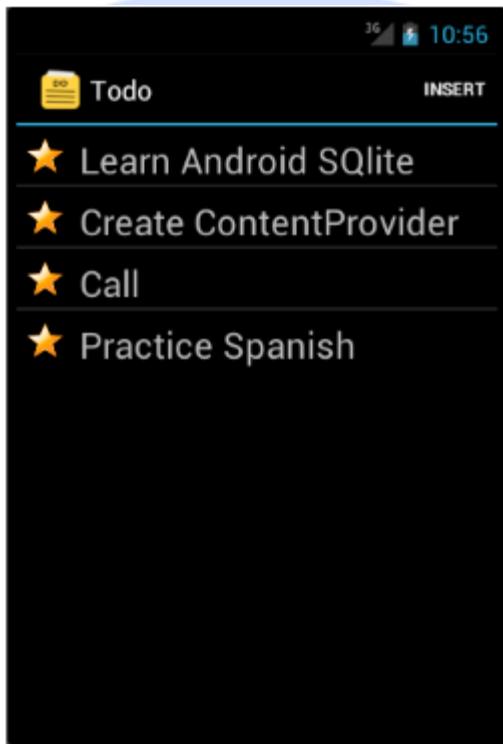


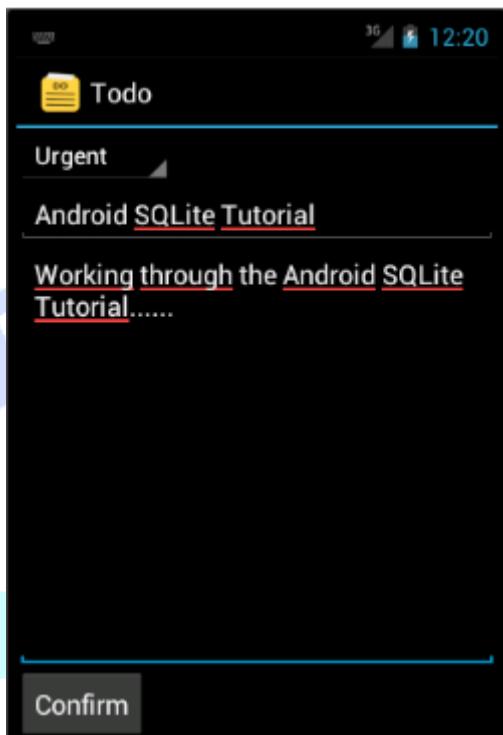
در بخش آموزشی حاضر یک اپلیکیشن به نام "To-do" ایجاد خواهید نمود. این برنامه به کاربر اجازه می دهد تا task یا کارهایی که باید انجام شوند را در لیستی وارد نماید. آیتم های ورودی در دیتابیس SQLite ذخیره شده و از طریق ContentProvider قابل دسترسی می باشد.

در نمونه اپلیکیشن جاری، کارها یا task هایی که داخل لیست درج می شوند، "todos" نام دارند. اپلیکیشن مورد نظر دارای دو activity می باشد که یکی برای مشاهده ی آیتم های todo و دیگری ویژه ی ایجاد و ویرایش آیتم مورد نظر تعییه شده است. دو activity این اپلیکیشن از طریق آبجکت های Intent با یکدیگر تعامل کرده و اطلاعات مورد نیاز را رد و بدل می نمایند.

به منظور بارگذاری و مدیریت Cursor به طور ناهمزمان (asynch)، اصلی اپلیکیشن از کلاس Loader بهره می‌گیرد.

در پایان اپلیکیشن دارای ظاهری مشابه زیر خواهد بود.





## ایجاد پروژه

یک پروژه و activity جدید به ترتیب به نام های de.vogella.android.todos و de.vogella.android.todos.database ایجاد نمایید. حال دیگری activity به نام TodoDetailActivity در پروژه TodosOverviewActivity در حال ایجاد نمایید.

## 4-5-1-Database-تعریف کلاس های مدیریت

پکیج de.vogella.android.todos.database را ایجاد نمایید. این پکیج کلاس هایی که عملیات مربوط به دیتابیس را اداره می کنند، در خود نگه می دارد.

همان طور که قبلاً گفته شد، بهتر است ویژه هر جدول در دیتابیس، یک کلاس جداگانه تعریف نمایید. هر چند نمونه اپلیکیشن حاضر یک جدول بیشتر ندارد، اما توصیه می شود از این روش

بپروردی کنید تا در صورت گسترش schema دیتابیس (پوشه ای که جداول دیتابیس در آن قرار می گیرند) تمہیدات لازم را از قبل اندیشیده و برای آن آماده باشد.

کلاس زیر را ایجاد نمایید. این کلاس مقادیر مربوط به اسم جدول و ستون های آن را در قالب چند constant نگه می دارد.

```
package de.vogella.android.todos.database;
import android.database.sqlite.SQLiteDatabase;
import android.util.Log;
public class TodoTable {
    // Database table
    public static final String TABLE_TODO = "todo";
    public static final String COLUMN_ID = "_id";
    public static final String COLUMN_CATEGORY = "category";
    public static final String COLUMN_SUMMARY = "summary";
    public static final String COLUMN_DESCRIPTION = "description";
    // Database creation SQL statement
    private static final String DATABASE_CREATE = "create table "
        + TABLE_TODO
        + "("
        + COLUMN_ID + " integer primary key autoincrement,"
        + COLUMN_CATEGORY + " text not null,"
        + COLUMN_SUMMARY + " text not null,"
        + COLUMN_DESCRIPTION
        + " text not null"
        + ");";
    public static void onCreate(SQLiteDatabase database) {
        database.execSQL(DATABASE_CREATE);
    }
    public static void onUpgrade(SQLiteDatabase database, int oldVersion,
        int newVersion) {
        Log.w(TodoTable.class.getName(), "Upgrading database from version "
            + oldVersion + " to " + newVersion
            + ", which will destroy all old data");
        database.execSQL("DROP TABLE IF EXISTS " + TABLE_TODO);
        onCreate(database);
    }
}
```

کلاس TodoTable زیر را ایجاد نمایید. این کلاس از SQLiteOpenHelper ارث برده و متدهای static کلاس کمکی (helper) فراخوانی می کند.

```
package de.vogella.android.todos.database;
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
public class TodoDatabaseHelper extends SQLiteOpenHelper {
    private static final String DATABASE_NAME = "todotable.db";
```

```

private static final int DATABASE_VERSION = 1;
public TodoDatabaseHelper(Context context) {
    super(context, DATABASE_NAME, null, DATABASE_VERSION);
}
// Method is called during creation of the database
@Override
public void onCreate(SQLiteDatabase database) {
    TodoTable.onCreate(database);
}
// Method is called during an upgrade of the database,
// e.g. if you increase the database version
@Override
public void onUpgrade(SQLiteDatabase database, int oldVersion,
        int newVersion) {
    TodoTable.onUpgrade(database, oldVersion, newVersion);
}
}

```

برای دسترسی به دیتابیس این بار مانند مثال قبلی از یک DAO (آبجکت دسترسی به داده) استفاده نخواهید نمود، بلکه یک ContentProvider جهت دسترسی به داده های مورد نیاز پیاده سازی خواهید کرد.

#### 4-5-2- ایجاد ContentProvider

ابتدا پکیج de.vogella.android.todos.contentprovider را ایجاد نمایید. سپس کلاس MyTodoContentProvider را ایجاد کنید که از ContentProvider ارث بری (extend) می کند.

```

package de.vogella.android.todos.contentprovider;
import java.util.Arrays;
import java.util.HashSet;
import android.content.ContentProvider;
import android.content.ContentResolver;
import android.content.ContentValues;
import android.content.UriMatcher;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteQueryBuilder;
import android.net.Uri;
import android.text.TextUtils;
import de.vogella.android.todos.database.TodoDatabaseHelper;
import de.vogella.android.todos.database.TodoTable;
public class MyTodoContentProvider extends ContentProvider {
    // database
    private TodoDatabaseHelper database;
    // used for the UriMatcher
    private static final int TODOS = 10;
    private static final int TODO_ID = 20;
    private static final String AUTHORITY = "de.vogella.android.todos.contentprovider";
}

```

```

private static final String BASE_PATH = "todos";
public static final Uri CONTENT_URI = Uri.parse("content://" + AUTHORITY
        + "/" + BASE_PATH);
public static final String CONTENT_TYPE = ContentResolver.CURSOR_DIR_BASE_TYPE
        + "/todos";
public static final String CONTENT_ITEM_TYPE = ContentResolver.CURSOR_ITEM_BASE_TYPE
        + "/todo";
private static final UriMatcher sURIMatcher = new UriMatcher(
        UriMatcher.NO_MATCH);
static {
    sURIMatcher.addURI(AUTHORITY, BASE_PATH, TODOS);
    sURIMatcher.addURI(AUTHORITY, BASE_PATH + "/#", TODO_ID);
}
@Override
public boolean onCreate() {
    database = new TodoDatabaseHelper(getContext());
    return false;
}
@Override
public Cursor query(Uri uri, String[] projection, String selection,
        String[] selectionArgs, String sortOrder) {
    // Using SQLiteQueryBuilder instead of query() method
    SQLiteQueryBuilder queryBuilder = new SQLiteQueryBuilder();
    // check if the caller has requested a column which does not exists
    checkColumns(projection);
    // Set the table
    queryBuilder.setTables(TodoTable.TABLE_TODO);
    int uriType = sURIMatcher.match(uri);
    switch (uriType) {
        case TODOS:
            break;
        case TODO_ID:
            // adding the ID to the original query
            queryBuilder.appendWhere(TodoTable.COLUMN_ID + "="
                    + uri.getLastPathSegment());
            break;
        default:
            throw new IllegalArgumentException("Unknown URI: " + uri);
    }
    SQLiteDatabase db = database.getWritableDatabase();
    Cursor cursor = queryBuilder.query(db, projection, selection,
            selectionArgs, null, null, sortOrder);
    // make sure that potential listeners are getting notified
    cursor.setNotificationUri(getContext().getContentResolver(), uri);
    return cursor;
}
@Override
public String getType(Uri uri) {
    return null;
}
@Override
public Uri insert(Uri uri, ContentValues values) {

```

```

int uriType = sURIMatcher.match(uri);
SQLiteDatabase sqlDB = database.getWritableDatabase();
long id = 0;
switch (uriType) {
case TODOS:
    id = sqlDB.insert(TodoTable.TABLE_TODO, null, values);
    break;
default:
    throw new IllegalArgumentException("Unknown URI: " + uri);
}
getContext().getContentResolver().notifyChange(uri, null);
return Uri.parse(BASE_PATH + "/" + id);
}

@Override
public int delete(Uri uri, String selection, String[] selectionArgs) {
    int uriType = sURIMatcher.match(uri);
    SQLiteDatabase sqlDB = database.getWritableDatabase();
    int rowsDeleted = 0;
    switch (uriType) {
    case TODOS:
        rowsDeleted = sqlDB.delete(TodoTable.TABLE_TODO, selection,
            selectionArgs);
        break;
    case TODO_ID:
        String id = uri.getLastPathSegment();
        if (TextUtils.isEmpty(selection)) {
            rowsDeleted = sqlDB.delete(
                TodoTable.TABLE_TODO,
                TodoTable.COLUMN_ID + "=" + id,
                null);
        } else {
            rowsDeleted = sqlDB.delete(
                TodoTable.TABLE_TODO,
                TodoTable.COLUMN_ID + "=" + id
                + " and " + selection,
                selectionArgs);
        }
        break;
    default:
        throw new IllegalArgumentException("Unknown URI: " + uri);
    }
    getContext().getContentResolver().notifyChange(uri, null);
    return rowsDeleted;
}

@Override
public int update(Uri uri, ContentValues values, String selection,
    String[] selectionArgs) {
    int uriType = sURIMatcher.match(uri);
    SQLiteDatabase sqlDB = database.getWritableDatabase();
    int rowsUpdated = 0;
    switch (uriType) {
    case TODOS:

```

```
rowsUpdated = sqldb.update(TodoTable.TABLE_TODO,
                           values,
                           selection,
                           selectionArgs);
        break;
    case TODO_ID:
        String id = uri.getLastPathSegment();
        if (TextUtils.isEmpty(selection)) {
            rowsUpdated = sqldb.update(TodoTable.TABLE_TODO,
                                       values,
                                       TodoTable.COLUMN_ID + "=" + id,
                                       null);
        } else {
            rowsUpdated = sqldb.update(TodoTable.TABLE_TODO,
                                       values,
                                       TodoTable.COLUMN_ID + "=" + id
                                       + " and "
                                       + selection,
                                       selectionArgs);
        }
        break;
    default:
        throw new IllegalArgumentException("Unknown URI: " + uri);
    }
    getContext().getContentResolver().notifyChange(uri, null);
    return rowsUpdated;
}

private void checkColumns(String[] projection) {
    String[] available = { TodoTable.COLUMN_CATEGORY,
                           TodoTable.COLUMN_SUMMARY, TodoTable.COLUMN_DESCRIPTION,
                           TodoTable.COLUMN_ID };
    if (projection != null) {
        HashSet<String> requestedColumns = new HashSet<String>(
            Arrays.asList(projection));
        HashSet<String> availableColumns = new HashSet<String>(
            Arrays.asList(available));
        // check if all columns which are requested are available
        if (!availableColumns.containsAll(requestedColumns)) {
            throw new IllegalArgumentException(
                "Unknown columns in projection");
        }
    }
}
```

MyTodoContentProvider توابع query(), insert(), update() و delete() را پیاده سازی می کند. این متدها به طور مستقیم از SQLiteInterface یا الگوی پیاده سازی SQLiteDatabase مشتق می شوند.

این کلاس همچنین با فراخوانی متodi به نام `checkColumns()`، اطمینان حاصل می کند که کوئری فقط و فقط ستون های معتبر و مجاز را درخواست می کند.

این اپلیکیشن خود را در فایل `AndroidManifest.xml` ثبت نمایید.

```
<application
    <!-- Place the following after the Activity
        Definition
    -->
<provider
    android:name=".contentprovider.MyTodoContentProvider"
    android:authorities="de.vogella.android.todos.contentprovider" >
</provider>
</application>
```

#### 4-5-3- منابع (Resource)

اپلیکیشن جاری به منابع و محتوای متعددی احتیاج دارد. ابتدا یک منوی `listmenu.xml` در پوشه `res/menu` تعریف نمایید. اگر از ویزارد Android resource برای ایجاد فایل `"listmenu.xml"` استفاده نمایید، پوشه `res` مورد نیاز به صورت خودکار برای شما ایجاد می شود. چنانچه فایل را خود به صورت دستی ایجاد کنید، در آن صورت پوشه را نیز باید خود بسازید.

این فایل XML برای تعریف ظاهر option menu در اپلیکیشن بکار می رود. با تنظیم مقدار `android:showAsAction="always"` آیتم مورد نظر در `ActionBar` اپلیکیشن به نمایش گذاشته می شود.

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
    <item
        android:id="@+id/insert"
        android:showAsAction="always"
        android:title="Insert">
    </item>
</menu>
```

کاربر می تواند برای آیتم ها اولویت قرار دهد. برای `priorities` یک آرایه ی رشته ای ایجاد خواهد نمود.

ابتدا فایل priority.xml زیر را در پوشه res/values ایجاد نمایید.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="priorities">
        <item>Urgent</item>
        <item>Reminder</item>
    </string-array>
</resources>
```

لازم است برای اپلیکیشن خود ثابت های رشته ای متعددی تعریف کنید. فایل strings.xml تحت res/values را ویرایش نمایید.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, Todo!</string>
    <string name="app_name">Todo</string>
    <string name="no.todos">Currently there are no Todo items maintained</string>
    <string name="menu_insert">Add Item</string>
    <string name="menu_delete">Delete Todo</string>
    <string name="todo_summary">Summary</string>
    <string name="todo_description">Delete Todo</string>
    <string name="todo_edit_summary">Summary</string>
    <string name="todo_edit_description">Description</string>
    <string name="todo_edit_confirm">Confirm</string>
</resources>
```

#### 4-5-4-تعریف فایل های layout

سه فایل layout تعریف خواهید کرد. یکی از فایل های layout برای نمایش و تنظیم ظاهر سطر در لیست مورد استفاده قرار می گیرد و دو فایل دیگر توسط activity های اپلیکیشن فراخوانی می شوند.

به آیکونی به نام reminder اشاره دارد. آیکونی از نوع "png" به نام drawable-hdpi, drawable-mdpi, drawable-ldpi در پوشه res/drawable در "reminder.png" ایجاد نمایید.

اگر خود آیکون مناسبی در دست ندارید، می توانید آیکونی که ویزارد اندروید (the res/drawable\* folders) در اختیار شما قرار می دهد را بکار ببرید یا اسم reference (reference) اشاره گر

به آن را در فایل layout ویرایش نمایید. لازم به ذکر است که ADT ممکن است اسم آیکون تولید شده را تغییر دهد. از اینرو اگر اسم فایل شما "ic\_launcher.png" نبود، تعجب نکنید.

در صورت تمایل می توانید تعریف آیکون را از فایل "todo\_row.xml" به کلی حذف کنید. این فایل را در مرحله ی بعدی ایجاد خواهید نمود.

فایل "todo\_row.xml" را در پوشه ی `res/layout` ایجاد نمایید.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >
    <ImageView
        android:id="@+id/icon"
        android:layout_width="30dp"
        android:layout_height="24dp"
        android:layout_marginLeft="4dp"
        android:layout_marginRight="8dp"
        android:layout_marginTop="8dp"
        android:src="@drawable/reminder" >
    </ImageView>
    <TextView
        android:id="@+id/label"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="6dp"
        android:lines="1"
        android:text="@+id/TextView01"
        android:textSize="24dp" >
    </TextView>
</LinearLayout>
```

حال فایل todo\_list.xml را ایجاد نمایید. این فایل ظاهر و چیدمان لیست را تعیین می کند.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <ListView
        android:id="@+id/list"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >
    </ListView>
```

```
<TextView
    android:id="@+id/empty"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/no.todos" />
</LinearLayout>
```

فایل todo\_edit.xml را ایجاد کنید. این فایل layout برای ویرایش و تنظیم ظاهر آیتم های فردی todo در TodoDetailActivity مورد استفاده قرار می گیرد.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <Spinner
        android:id="@+id/category"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:entries="@array/priorities" >
    </Spinner>
    <LinearLayout
        android:id="@+id/LinearLayout01"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >
        <EditText
            android:id="@+id/todo_edit_summary"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:hint="@string/todo_edit_summary"
            android:imeOptions="actionNext" >
        </EditText>
    </LinearLayout>
    <EditText
        android:id="@+id/todo_edit_description"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:gravity="top"
        android:hint="@string/todo_edit_description"
        android:imeOptions="actionNext" >
    </EditText>
    <Button
        android:id="@+id/todo_edit_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/todo_edit_confirm" >
    </Button>
```

```
</LinearLayout>
```

## ۴-۵-۵- تنظیم ها Activity

کد های خود را به صورت زیر ویرایش کنید. ویرایش را از فایل `TodosOverviewActivity.java` آغاز نمایید.

```
package de.vogella.android.todos;
import android.app.ListActivity;
import android.app.LoaderManager;
import android.content.CursorLoader;
import android.content.Intent;
import android.content.Loader;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.view.ContextMenu;
import android.view.ContextMenu.ContextMenuItemInfo;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView.AdapterContextMenuInfo;
import android.widget.ListView;
import android.widget.SimpleCursorAdapter;
import de.vogella.android.todos.contentprovider.MyTodoContentProvider;
import de.vogella.android.todos.database.TodoTable;
/*
 * TodosOverviewActivity displays the existing todo items
 * in a list
 *
 * You can create new ones via the ActionBar entry "Insert"
 * You can delete existing ones via a long press on the item
 */
public class TodosOverviewActivity extends ListActivity implements
    LoaderManager.LoaderCallbacks<Cursor> {
    private static final int ACTIVITY_CREATE = 0;
    private static final int ACTIVITY_EDIT = 1;
    private static final int DELETE_ID = Menu.FIRST + 1;
    // private Cursor cursor;
    private SimpleCursorAdapter adapter;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.todo_list);
        this.getListView().setDividerHeight(2);
        fillData();
    }
}
```

```

        registerForContextMenu(getListView());
    }
    // create the menu based on the XML defintion
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.listmenu, menu);
        return true;
    }
    // Reaction to the menu selection
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
        case R.id.insert:
            createTodo();
            return true;
        }
        return super.onOptionsItemSelected(item);
    }
    @Override
    public boolean onContextItemSelected(MenuItem item) {
        switch (item.getItemId()) {
        case DELETE_ID:
            AdapterContextMenuInfo info = (AdapterContextMenuInfo) item
                .getMenuInfo();
            Uri uri = Uri.parse(MyTodoContentProvider.CONTENT_URI + "/"
                + info.id);
            getContentResolver().delete(uri, null, null);
            fillData();
            return true;
        }
        return super.onContextItemSelected(item);
    }
    private void createTodo() {
        Intent i = new Intent(this, TodoDetailActivity.class);
        startActivity(i);
    }
    // Opens the second activity if an entry is clicked
    @Override
    protected void onListItemClick(ListView l, View v, int position, long id) {
        super.onListItemClick(l, v, position, id);
        Intent i = new Intent(this, TodoDetailActivity.class);
        Uri todoUri = Uri.parse(MyTodoContentProvider.CONTENT_URI + "/" + id);
        i.putExtra(MyTodoContentProvider.CONTENT_ITEM_TYPE, todoUri);
        startActivity(i);
    }
    private void fillData() {
        // Fields from the database (projection)
        // Must include the _id column for the adapter to work
        String[] from = new String[] { TodoTable.COLUMN_SUMMARY };
        // Fields on the UI to which we map
        int[] to = new int[] { R.id.label };
    }
}

```

```

        getLoaderManager().initLoader(0, null, this);
        adapter = new SimpleCursorAdapter(this, R.layout.todo_row, null, from,
                to, 0);
        setListAdapter(adapter);
    }

    @Override
    public void onCreateContextMenu(ContextMenu menu, View v,
            ContextMenuItem menuInfo) {
        super.onCreateContextMenu(menu, v, menuInfo);
        menu.add(0, DELETE_ID, 0, R.string.menu_delete);
    }

    // creates a new loader after the initLoader () call
    @Override
    public Loader<Cursor> onCreateLoader(int id, Bundle args) {
        String[] projection = { TodoTable.COLUMN_ID, TodoTable.COLUMN_SUMMARY };
        CursorLoader cursorLoader = new CursorLoader(this,
                MyTodoContentProvider.CONTENT_URI, projection, null, null, null);
        return cursorLoader;
    }

    @Override
    public void onLoadFinished(Loader<Cursor> loader, Cursor data) {
        adapter.swapCursor(data);
    }

    @Override
    public void onLoaderReset(Loader<Cursor> loader) {
        // data is not available anymore, delete reference
        adapter.swapCursor(null);
    }
}

```

سپس فایل `TodoDetailActivity.java` را به صورت زیر ویرایش کنید.

```

package de.vogella.android.todos;
import android.app.Activity;
import android.content.ContentValues;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.Toast;
import de.vogella.android.todos.contentprovider.MyTodoContentProvider;
import de.vogella.android.todos.database.TodoTable;
/*
 * TodoDetailActivity allows to enter a new todo item
 * or to change an existing

```

```

/*
public class TodoDetailActivity extends Activity {
    private Spinner mCategory;
    private EditText mTitleText;
    private EditText mBodyText;
    private Uri todoUri;
    @Override
    protected void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        setContentView(R.layout.todo_edit);
        mCategory = (Spinner) findViewById(R.id.category);
        mTitleText = (EditText) findViewById(R.id.todo_edit_summary);
        mBodyText = (EditText) findViewById(R.id.todo_edit_description);
        Button confirmButton = (Button) findViewById(R.id.todo_edit_button);
        Bundle extras = getIntent().getExtras();
        // check from the saved Instance
        todoUri = (bundle == null) ? null : (Uri) bundle
            .getParcelable(MyTodoContentProvider.CONTENT_ITEM_TYPE);
        // Or passed from the other activity
        if (extras != null) {
            todoUri = extras
                .getParcelable(MyTodoContentProvider.CONTENT_ITEM_TYPE);
            fillData(todoUri);
        }
        confirmButton.setOnClickListener(new View.OnClickListener() {
            public void onClick(View view) {
                if (TextUtils.isEmpty(mTitleText.getText().toString())) {
                    makeToast();
                } else {
                    setResult(RESULT_OK);
                    finish();
                }
            }
        });
    }
    private void fillData(Uri uri) {
        String[] projection = { TodoTable.COLUMN_SUMMARY,
            TodoTable.COLUMN_DESCRIPTION, TodoTable.COLUMN_CATEGORY };
        Cursor cursor = getContentResolver().query(uri, projection, null, null);
        if (cursor != null) {
            cursor.moveToFirst();
            String category = cursor.getString(cursor
                .getColumnIndexOrThrow(TodoTable.COLUMN_CATEGORY));
            for (int i = 0; i < mCategory.getCount(); i++) {
                String s = (String) mCategory.getItemAtPosition(i);
                if (s.equalsIgnoreCase(category)) {
                    mCategory.setSelection(i);
                }
            }
            mTitleText.setText(cursor.getString(cursor
                .getColumnIndexOrThrow(TodoTable.COLUMN_SUMMARY)));
        }
    }
}

```

```

        mBodyText.setText(cursor.getString(cursor
            .getColumnIndexOrThrow(TodoTable.COLUMN_DESCRIPTION)));
        // always close the cursor
        cursor.close();
    }
}

protected void onSaveInstanceState(Bundle outState) {
    super.onSaveInstanceState(outState);
    saveState();
    outState.putParcelable(MyTodoContentProvider.CONTENT_ITEM_TYPE, todoUri);
}

@Override
protected void onPause() {
    super.onPause();
    saveState();
}

private void saveState() {
    String category = (String) mCategory.getSelectedItem();
    String summary = mTitleText.getText().toString();
    String description = mBodyText.getText().toString();
    // only save if either summary or description
    // is available
    if (description.length() == 0 && summary.length() == 0) {
        return;
    }
    ContentValues values = new ContentValues();
    values.put(TodoTable.COLUMN_CATEGORY, category);
    values.put(TodoTable.COLUMN_SUMMARY, summary);
    values.put(TodoTable.COLUMN_DESCRIPTION, description);
    if (todoUri == null) {
        // New todo
        todoUri = getContentResolver().insert(
            MyTodoContentProvider.CONTENT_URI, values);
    } else {
        // Update todo
        getContentResolver().update(todoUri, values, null, null);
    }
}
private void makeToast() {
    Toast.makeText(TodoDetailActivity.this, "Please maintain a summary",
        Toast.LENGTH_LONG).show();
}
}

```

محتوای فایل AndroidManifest.xml در نهایت به صورت زیر خواهد بود.

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="de.vogella.android.todos"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk android:minSdkVersion="15" />

```

```

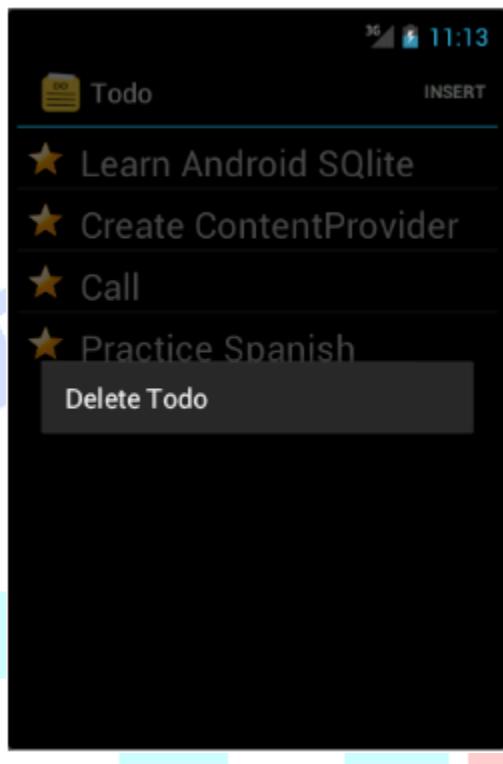
<application
    android:icon="@drawable/icon"
    android:label="@string/app_name" >
    <activity
        android:name=".TodosOverviewActivity"
        android:label="@string/app_name" >
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity
        android:name=".TodoDetailActivity"           android:windowSoftInputMode="stateVisible|adjustResize"
    >
    </activity>
    <provider     android:name=".contentprovider.MyTodoContentProvider"
    android:authorities="de.vogella.android.todos.contentprovider" >
        </provider>
    </application>
</manifest>

```

## تست اپلیکیشن

اپلیکیشن خود را راه اندازی کنید. با کلیک بر روی دکمه **Insert** در ActionBar طبیعتا باید بتوانید آیتم جدیدی را در لیست درج نمایید.

جهت حذف آیتم دلخواه از لیست کافی است بر روی آن طولانی مدت کلیک نمایید.



به منظور ویرایش آیتم جاری در لیست، بر روی سطر میزبان آن کلیک نمایید. با این کار دوم راه اندازی می شود.

#### 6-5-4- محل ذخیره سازی دیتابیس SQLite

دیتابیس را همراه با تمامی اطلاعات آن داخل یک فایل ذخیره می کند. در صورتی که به این فایل دسترسی دارید، می توانید به طور مستقیم با دیتابیس کار کنید. دسترسی مستقیم به فایل مجبور تنها در محیط شبیه ساز و دستگاه های root شده امکان پذیر می باشد.

یک دستگاه استاندارد اندروید اجازه دسترسی و خواندن (read-access) با فایل دیتابیس را نمی دهد.

## 4-5-7- دسترسی به دیتابیس از راه دور به وسیله‌ی command line (Shell access)

در محیط شبیه ساز یا دستگاه root شده اندروید، می‌توان از طریق پنجره‌ی فرمان یا command line به دیتابیس SQLite دسترسی داشت. جهت اتصال به دستگاه کافی است دستور زیر را در پنجره‌ی فرمان اجرا کنید.

adb shell

در پوشه‌ی نصب "platform-tools" و زیرپوشه‌ی "Command adb" باشد.

سپس دستور "cd" را برای تغییر پوشه‌ی دیتابیس فراخوانی نموده و با اجرای دستور "sqlite3" به دیتابیس متصل شوید. برای مثال در اپلیکیشن جاری دستورات لازم به شرح زیر است:

```
# Switch to the data directory
cd /data/data
# Our application
cd de.vogella.android.todos
# Switch to the database dir
cd databases
# Check the content
ls
# Assuming that there is a todotable file
# connect to this table
sqlite3 todotable.db
```

پرکاربرد ترین دستورات SQLite به شرح زیر می‌باشند:

دستور	شرح
.help	تمامی دستورات و آیشن‌ها را لیست می‌کند.
.exit	از دستور sqlite3 خارج می‌شود.
.schema	دستورات CREATE که برای ایجاد جداول دیتابیس جاری فراخوانی شدند را لیست می‌کند.

## بهینه سازی اپلیکیشن

تغییراتی که در SQLite اعمال می شوند همگی ACID هستند (Atomicity: تراکنش ها باید کامل اجرا شوند. Consistency: تمامی تراکنش ها باید دیتابیس را از یک حالت کامل و قابل قبول به یک حالت بی عیب و کامل دیگر منتقل کند. Isolation: تراکنش هایی که موازی اجرا می شوند باید کاملا از هم مجزا بوده و در جامعیت دیتابیس اختلالی ایجاد نکند. Durability: کلیه ی تغییراتی که تراکنش ها بر روی دیتابیس اعمال می کنند همگی باید ذخیره شده و در صورت رخداد خطای ناگهانی آن ها را به حالت اول بازگرداند). در نتیجه تمامی عملیات insert, update و delete ای که بر روی پایگاه داده اجرا می شوند، از این چهار ویژگی برخوردار بوده و از آن ها تبعیت می کنند. در پی این ویژگی، به ناگزیر مقداری overhead و سربار در پردازش های دیتابیس وارد شده و کارایی را پایین می آورد. به همین جهت توصیه می شود عملیات update در قالب یک تراکنش (transaction) گنجانده (wrap) و پس از انجام موفقیت آمیز تمامی عملیات لازم تراکنش را ثبت نهایی (commit) نمایید. با این اقدام سرعت و کارایی اپلیکیشن به طور چشم گیری افزایش می یابد.

کد زیر این افزایش کارایی را به نمایش می گذارد.

```
db.beginTransaction();
try {
    for (int i = 0; i < values.length; i++) {
        // TODO prepare ContentValues object values
        db.insert(your_table, null, values);
        // In case you do larger updates
        yieldIfContendedSafely()
    }
    db.setTransactionSuccessful();
} finally {
    db.endTransaction();
}
```

برای آپدیت های زیاد می بایست متدهای SQLite را فراخوانی نمایید. در طول اجرای تراکنش بر روی دیتابیس قفل می گذارد. حال اگر سرویس گیرنده ی دیگری از داده

های مورد نظر کوئری بگیرد، با فراخوانی این متاد، اندروید تراکنش را متوقف کرده و یک تراکنش جدید باز می کند. از این طریق فرایند دوم می تواند به داده ها دسترسی پیدا کند.



## بخش دوم :

# پردازش فایل های XML در اندروید با استفاده از تحلیلگر نحوی xmlPullParser (Parser)

مبحث حاضر به شرح نحوه ی پردازش فایل های XML در اندروید می پردازد.

## 6-4-پردازش فایل های XML با استفاده از تحلیلگر نحوی XmlPullParser

زبان برنامه نویسی Java تعدادی کتابخانه ی استاندارد برای پردازش فایل های XML ارائه می دهد. تحلیلگرهای نحوی SAX و DOM XML که در محیط Android نیز قابل استفاده می باشند.

API یا توابع تحلیلگر نحوی SAX و DOM در زبان جاوا و محیط Android یکسان هستند. البته لازم به توضیح است که دو API نام بردۀ محدودیت های خودشان را دارند و برای استفاده در اندروید نمی شوند. به این دلیل در آموزش جاری مثالی از کاربرد این کتابخانه عنوان نخواهد توصیه شد.

Java همچنین parser یا تحلیل گر نحوی Stax را ارائه می دهد که بین جاوا و بستر اجرای اندروید (platform) مشترک نیست.

محیط Android برای تحلیل گرامری (parse) و نوشتن در فایل XML از کلاس xmlPullParser استفاده می کند. این parser به گونه ای معادل Stax در جاوا است (اما جزئی از بستر اجرای جاوا نمی باشد). برای دسترسی به xmlPullParser می توانید به Java SE آدرس مراجعه نمایید. آدرس <http://www.xmlpull.org/>

جهت تحلیل گرامری و نوشتن در فایل های XML در محیط اجرای اندروید (Android Platform)، API کلاس XmlPullParser توصیه می شود. این کتابخانه در مقایسه با SAX و DOM از توابع و

ساده تری برخوردار بوده، سرعت اجرای آن بالا است و نسبت به DOM API حافظه‌ی کمتری را اشغال می‌کند.

## مثالی از کاربرد XmlPullParser

این کتابخانه یک مثال کاربردی و مناسب در خصوص نحوه‌ی استفاده از آن ارائه می‌دهد (javadoc) از توضیحات و مستندات نوشته در java source فایل HTML ایجاد می‌کند و برای دیگر توسعه‌دهنگان این امکان را فراهم می‌کند تا کد شما بهتر بفهمند.

```
import java.io.IOException;
import java.io.StringReader;
import org.xmlpull.v1.XmlPullParser;
import org.xmlpull.v1.XmlPullParserException.html;
import org.xmlpull.v1.XmlPullParserFactory;
public class SimpleXmlPullParserApp
{
    public static void main (String args[])
        throws XmlPullParserException, IOException
    {
        XmlPullParserFactory factory = XmlPullParserFactory.newInstance();
        factory.setNamespaceAware(true);
        XmlPullParser xpp = factory.newPullParser();
        xpp.setInput( new StringReader( "<foo>Hello World!</foo>" ) );
        int eventType = xpp.getEventType();
        while (eventType != XmlPullParser.END_DOCUMENT) {
            if(eventType == XmlPullParser.START_DOCUMENT) {
                System.out.println("Start document");
            } else if(eventType == XmlPullParser.END_DOCUMENT) {
                System.out.println("End document");
            } else if(eventType == XmlPullParser.START_TAG) {
                System.out.println("Start tag "+xpp.getName());
            } else if(eventType == XmlPullParser.END_TAG) {
                System.out.println("End tag "+xpp.getName());
            } else if(eventType == XmlPullParser.TEXT) {
                System.out.println("Text "+xpp.getText());
            }
            eventType = xpp.next();
        }
    }
}
```



388

آدرس آموزشگاه : تهران - خیابان شریعتی - بالاتر از خیابان ملک - جنب بانک صادرات - پلاک 651 طبقه دوم - واحد 7

88146323 - 88446780 - 88146330

### بخش سوم :

## ذخیره ی ماندگار داده ها در اپلیکیشن اندروید با استفاده از preferences API (ذخیره و بازیابی اطلاعات مربوط به تنظیمات کاربر) و File API

این آموزش تمرکز خود را بر شرح نحوه ی ذخیره سازی جفت های کلید-مقدار در فایل های اندروید (File based persistence) با فراخوانی preference API قرار می دهد. سپس چگونگی خواندن/نوشتن فایل در اندروید را تشریح می کند.



### 7-4-1 روشهای ذخیره ی ماندگار داده ها به صورت محلی (local data persistence)

اندروید این امکان را فراهم می آورد تا داده های اپلیکیشن را در سیستم فایل ذخیره نمایید. در واقع سیستم اندروید به ازای هر اپلیکیشن یک پوشه ی [data/data/[application package]] ایجاد می کند.

اندروید برای ذخیره ی داده ها در سیستم فایل محلی روش های زیر را پشتیبانی می کند:

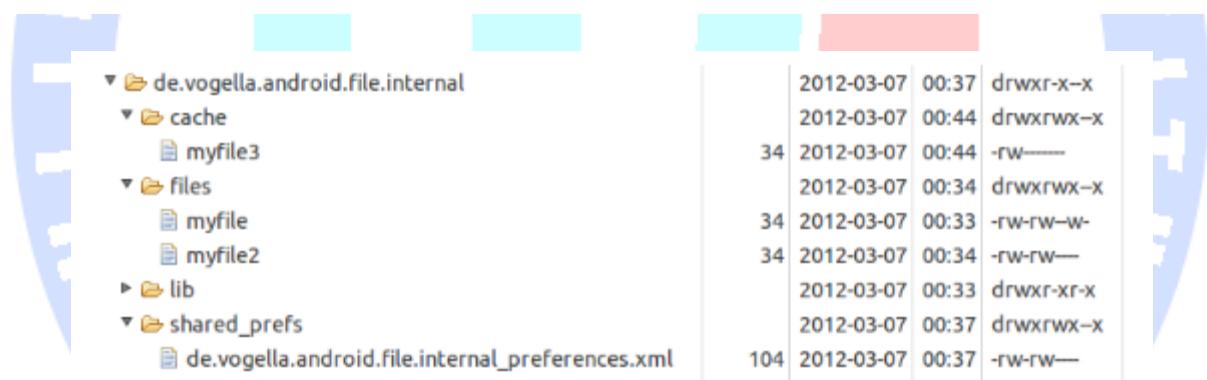
- Files – می توان داده ها را در فایل ذخیره کرده و بروز رسانی نمود (روشی قدیمی).
- Preferences – اندروید به شما اجازه می دهد جفت های کلید-مقدار از نوع داده ای اولیه (primitive data type) را به صورت دائمی ذخیره نمایید (به شما امکان می دهد اطلاعات مربوط به تنظیمات انتخابی کاربر را با استفاده از آبجکت SharedPreferences به صورت کلید-مقدار ذخیره کنید).

• می توان نمونه یا نسخه های موجود از دیتابیس SQLite database داده های اپلیکیشن را در سیستم فایل و به صورت محلی ذخیره کرد (ذخیره ی داده های اپلیکیشن در سیستم مدیریت متعددالشكل دیتابیس که اندروید توسط SQLite پشتیبانی می کند).

فایل ها در پوشه ی files و تنظیمات اپلیکیشن در قالب فایل های XML داخل پوشه ی shared-prefs ذخیره و نگهداری می شوند.

اگر اپلیکیشن شما یک دیتابیس SQLite ایجاد کند، اندروید به صورت پیش فرض این دیتابیس را در دایرکتوری اصلی اپلیکیشن و تحت پوشه ی databases ذخیره می نماید.

تصویر زیر یک سیستم فایل را به نمایش می گذارد که فایل های ساده، فایل های کش شده (cache) و اطلاعات مربوط به وضعیت تنظیمات انتخابی کاربر (preferences) را شامل می شود.



	Date	Time	Permissions	
de.vogella.android.file.internal	2012-03-07	00:37	drwxr-x-x	
cache	2012-03-07	00:44	drwxrwx-x	
myfile3	34	2012-03-07	00:44	-rw----
files				drwxrwx-x
myfile	34	2012-03-07	00:34	-rw-rw-w-
myfile2	34	2012-03-07	00:34	-rw-rw--
lib				drwxr-xr-x
shared_prefs	104	2012-03-07	00:37	drwxrwx-x
de.vogella.android.file.internal_preferences.xml				-rw-rw--

تنها اپلیکیشن است که اجازه ی درج داده و نوشتمن در پوشه ی اصلی اپلیکیشن (app-directory) را دارد. اپلیکیشن همچنین این قابلیت را دارد که زیرپوشه های (sub-directory) لازم را در پوشه ی اصلی (application directory) ایجاد نماید. جهت ایجاد این زیرپوشه ها، اپلیکیشن مورد نظر می تواند مجوزهای لازم در سطح خواندن و نوشتمن را به دیگر برنامه ها اعطا کند.

## 4-7-2- ذخیره سازی internal داده در مقایسه با ذخیره های داده ها به صورت external

اندروید دارای دو حافظه های داخلی و خارجی است و داده ها را به صورت internal و external ذخیره می کند. حافظه های خارجی دارای سطح دسترسی شخصی نبوده (private نیست) و همیشه در دسترس نیست. به عنوان مثال، زمانی که دستگاه اندروید از طریق USB به کامپیوتر وصل می شود، این حافظه های خارجی به طور موقت از دسترس اپلیکیشن های اندروید خارج می گردد.

## 4-7-3- جایگذاری اپلیکیشن در حافظه های خارجی

از ویرایش 8 مجموعه ابزار ساخت و توسعه های اندروید (SDK level 8) می توان در فایل تنظیمات (manifest) مشخص نمود که اپلیکیشن مورد نظر امکان نصب بر روی حافظه های خارجی را داشته باشد یا اینکه اپلیکیشن مجبور بایستی بر روی حافظه های خارجی جایگذاری شود. برای نیل به این هدف کافی است مقدار android:installLocation را در فایل تنظیمات برابر preferExternal یا auto قرار دهید.

در این شرایط، ممکن است برخی از کامپوننت های نرم افزاری و تشکیل دهنده اپلیکیشن بر روی mount point یا محل ذخیره سازی رمزگاری شده (encrypted) خارجی جایگذاری شوند. دیتابیس و سایر داده های private (با سطح دسترسی شخصی) همچنان فقط بر روی حافظه های داخلی سیستم (storage system) ذخیره می شوند.

## 8-4- Preferences (ذخیره و بازگردانی اطلاعات مربوط به تنظیمات کاربر)

### 4-8-1- ذخیره جفت های کلید-مقدار از نوع داده ای اولیه

اندروید با ارائه و پشتیبانی از کلاس SharedPreferences این امکان را مهیا می کند تا جفت های کلید-مقدار از جنس داده ای اولیه را در سیستم فایل اندروید ذخیره نمایید (تنظیمات انتخابی کاربر همچون اندازه های فونت و رنگ متن که داده های کوچک هستند را به صورت key-value ذخیره نمایید).

## Preferences و تنظیمات انتخابی کاربر را می توان به راحتی داخل فایل های منبع XML تعریف نمود.

کلاس PreferenceManager با ارائه ی توابعی به توسعه دهنده این امکان را می دهد تا اطلاعات مربوط به تنظیمات کاربر، مستقر در یک فایل XML را بازیابی کند. کد زیر نحوه ی دسترسی به تنظیمات ذخیره شده در یک فایل XML را به نمایش می گذارد.

```
# getting preferences from a specified file
SharedPreferences settings = getSharedPreferences("Test", Context.MODE_PRIVATE);
```

Preferences می بایست به صورت private ویژه ی اپلیکیشن مورد نظر ایجاد شود. تمامی کامپوننت ها و اجزا تشکیل دهنده ی اپلیکیشن می توانند به اطلاعات preferences دسترسی داشته باشند. به اشتراک گذاری اطلاعات با یک اپلیکیشن دیگر که دارای فایل preference سراسری با دسترسی در سطح خواندن و نوشتن (world readable/writable) می باشد، امری است که به ندرت اتفاق می افتد. چرا که در این صورت کامپوننت خارجی می بایست از اسم و محل قرار گیری دقیق فایل مطلع باشد.

متند Preferences پیش فرض را به راحتی فراخوانی با مقدار preference کامپوننت های اپلیکیشن قرار می گیرد. برای دسترسی به مقدار preference کافی است کلید مربوطه و آبجکتی از کلاس SharedPreferences را مانند زیر مورد استفاده قرار دهید.

```
SharedPreferences settings = PreferenceManager.getDefaultSharedPreferences(getApplicationContext());
String url = settings.getString("url", "n/a");
```

به منظور ایجاد و ویرایش مقادیر preferences لازم است متند edit() را بر روی آبجکت SharedPreferences فراخوانی نمایید. پس از ویرایش مقدار، بایستی متند apply() را فراخوانی کنید تا مقادیر جدید به صورت ناهمzman در سیستم فایل اعمال شوند. لازم به توضیح است که متند commit() تغییرات را به صورت همزمان به فایل سیستم اعمال می کند، از این جهت استفاده از متند مذکور در این سناریو توصیه نمی شود.

```
Editor edit = preferences.edit();
edit.putString("username", "new_value_for_user");
edit.apply();
```

## 4-8-2- گوش فراخوانی به تغییرات در تنظیمات کاربر به وسیله ی preference listener

می توانید با فراخوانی متد (`registerOnSharedPreferenceChangeListener()`) بر روی آبجکت `SharedPreference` به تغییرات گوش داده و از آن ها مطلع شوید.

```
SharedPreferences prefs =
    PreferenceManager.getDefaultSharedPreferences(this);
// Instance field for listener
listener = new SharedPreferences.OnSharedPreferenceChangeListener() {
    public void onSharedPreferenceChanged(SharedPreferences prefs, String key) {
        // Your Implementation
    }
};

prefs.registerOnSharedPreferenceChangeListener(listener);
```

توجه داشته باشید که `SharedPreferences` تمامی `listener` یا گوش فراخوان ها را داخل `WeakHashMap` نگه می دارد، به همین علت چنانچه کد شما اشاره گری به آن تعریف نکرده باشد (آن را داخل متغیر نگه ندارد)، `listener` ها بازیافت می شوند و اطلاعات مورد نظر شما از دست خواهد رفت.

### آموزش: آنچه لازم است

پروژه‌ی این بخش مبتنی بر نمونه اپلیکیشن "de.vogella.android.socialapp" از آموزش `action bar` در مباحث قبلی می باشد.

### تمرین: آنچه لازم است

برای تمرین زیر لازم است یک پروژه‌ی اندرویدی به نام `com.example.android.rssfeed` همراه با یک المان `action bar` در `Settings` ایجاد نمایید.

### تمرین: ذخیره ی تنظیمات کاربر برای RSSfeed

در تمرین جاری، برنامه را گونه ای می نویسید که یک activity دیگر باز شده و به کاربر این امکان را بدهد تا URL دلخواه خود برای RSS feed را وارد نماید. این activity ظاهر و UI خود را از یک فایل xml که حاوی تنظیمات انتخابی کاربر است (xml preference file)، می خواند.

### 4-8-3-ایجاد فایل preference

یک فایل XML به نام XML mypreferences.xml در پوشه‌ی XML ایجاد نمایید. المان‌های زیر را به آن اضافه نمایید.

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android" >
    <EditTextPreference
        android:key="url"
        android:title="Rss feed URL"
        android:inputType="textUri"/>
    <CheckBoxPreference android:title="Aktiv" android:key="active"/>
</PreferenceScreen>
```

### 4-8-4-ایجاد activity جهت دریافت ورودی از کاربر

ابتدا کلاس SettingsActivity که از PreferenceActivity ارث بری می کند را ایجاد نمایید. این فایل activity را با رگذاری نموده و به کاربر اجازه می دهد تا مقادیر را ویرایش کند.

```
package com.example.android.rssreader;
import android.os.Bundle;
import android.preference.PreferenceActivity;
public class SettingsActivity extends PreferenceActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        addPreferencesFromResource(R.xml.mypreferences);
    }
}
```

این کلاس را به عنوان یک activity در فایل تنظیمات اپلیکیشن AndroidManifest.xml ثبت و معرفی نمایید.

### 4-8-5-متصل کردن activity مربوط به تنظیمات و دریافت ورودی کاربر

activity مربوط به preference و دریافت تنظیمات دلخواه کاربر را با فراخوانی متد onOptionsItemSelected() باز نمایید.

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
```

```

MenuInflater inflater = getMenuInflater();
inflater.inflate(R.menu.mainmenu, menu);
return true;
}
// This method is called once the menu is selected
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.preferences:
            // Launch settings activity
            Intent i = new Intent(this, SettingsActivity.class);
            startActivity(i);
            break;
            // more code...
    }
    return true;
}

```

## 8-4-بارگذاری RSS feed با استفاده از مقدار preference

تکه کد زیر نشان می دهد چگونه می توان از یک متند به مقدار preference دسترسی پیدا کرد.

```

// if you use this in a service or activity you can use this
// if you use this in a fragment use getActivity() or getContent() as parameter
SharedPreferences settings = PreferenceManager.getDefaultSharedPreferences(this);
String url = settings.getString("url", "http://www.vogella.com/article.rss");

```

## تست اپلیکیشن

اپلیکیشن خود را اجرا نمایید. آیتم Settings (action bar) را از action bar انتخاب کنید. بایستی یک دیگر به نمایش در آمده و شما بتوانید آدرس URL دلخواه را وارد کنید. پس از کلیک بر روی دکمه ی بازگشت (back) و بروز رسانی (refresh)، با بررسی اطمینان حاصل کنید که مقدار url در activity بکار برده شده است.

## تمرین اضافی: نمایش مقدار جاری در settings

تکه کد زیر نشان می دهد چگونه می توان مقدار جاری را در صفحه ی preference نمایش داد.

```

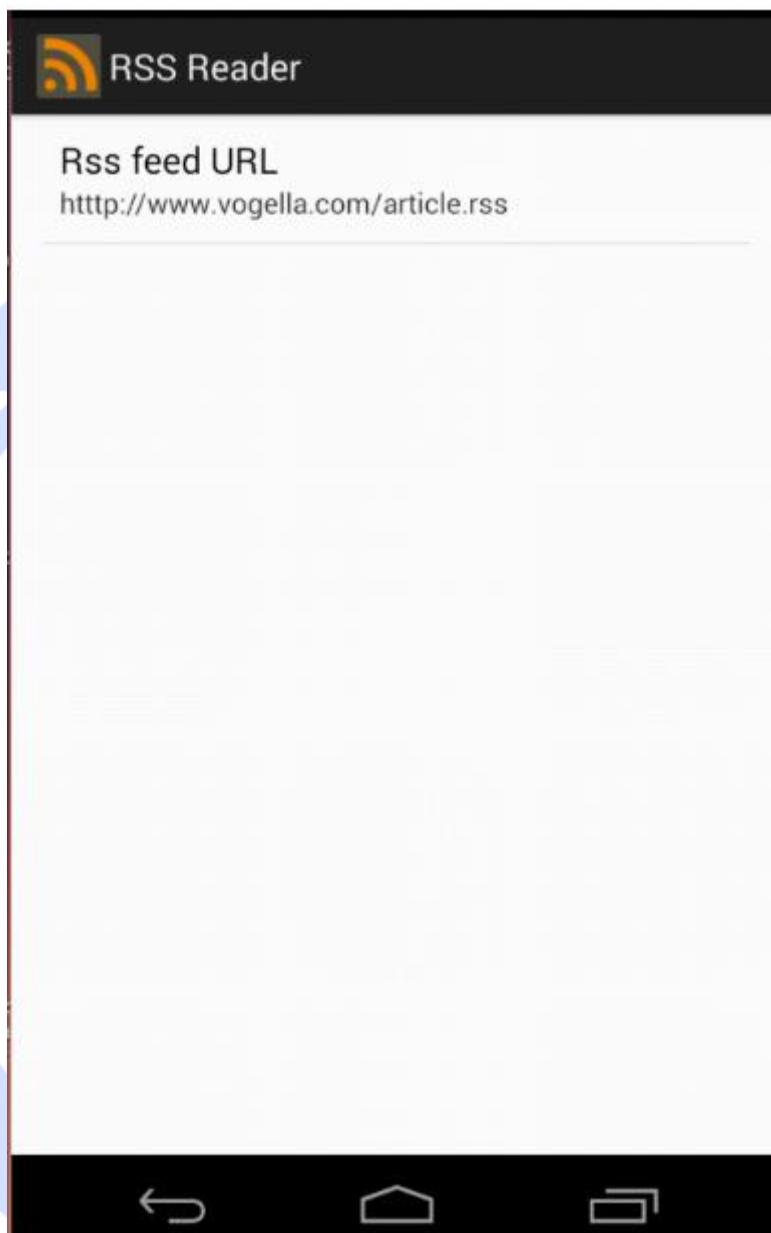
package com.example.android.rssreader;
import android.content.SharedPreferences;

```

```

import android.content.SharedPreferences.OnSharedPreferenceChangeListener;
import android.os.Bundle;
import android.preference.EditTextPreference;
import android.preference.ListPreference;
import android.preference.Preference;
import android.preference.PreferenceActivity;
import android.preference.PreferenceCategory;
public class SettingsActivity extends PreferenceActivity implements
    OnSharedPreferenceChangeListener {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        addPreferencesFromResource(R.xml.mypreferences);
        // show the current value in the settings screen
        for (int i = 0; i < getPreferenceScreen().getPreferenceCount(); i++) {
            initSummary(getPreferenceScreen().getPreference(i));
        }
    }
    @Override
    protected void onResume() {
        super.onResume();
        getPreferenceScreen().getSharedPreferences()
            .registerOnSharedPreferenceChangeListener(this);
    }
    @Override
    protected void onPause() {
        super.onPause();
        getPreferenceScreen().getSharedPreferences()
            .unregisterOnSharedPreferenceChangeListener(this);
    }
    @Override
    public void onSharedPreferenceChanged(SharedPreferences sharedpreferences,
        String key) {
        updatePreferences(findPreference(key));
    }
    private void initSummary(Preference p) {
        if (p instanceof PreferenceCategory) {
            PreferenceCategory cat = (PreferenceCategory) p;
            for (int i = 0; i < cat.getPreferenceCount(); i++) {
                initSummary(cat.getPreference(i));
            }
        } else {
            updatePreferences(p);
        }
    }
    private void updatePreferences(Preference p) {
        if (p instanceof EditTextPreference) {
            EditTextPreference editTextPref = (EditTextPreference) p;
            p.setSummary(editTextPref.getText());
        }
    }
}

```



## File API-9-4

### 4-9-1 استفاده از File API

برای دسترسی به سیستم فایل می توانید از طریق کلاس های `java.io` اقدام نمایید. اندروید کلاس های کمکی (`helper`) در اختیار توسعه دهنده قرار می دهد که ایجاد و دسترسی به فایل ها و

پوشه های جدید را آسان می سازد. برای مثال، `getDir(String, int)` پوشه `s` مورد نیاز را ایجاد نموده و دسترسی به آن را فراهم می سازد. متدهای `openFileInput(String s)` یک فایل برای ورودی باز کرده و متدهای `openFileOutput(String s, Context.MODE_PRIVATE)` نیز یک فایل جدید ایجاد می نماید.

اندروید از فایل های خواندنی و نوشتنی سراسری (`world readable/writable`) پشتیبانی می کند. اما به طور کلی توصیه می شود دسترسی به فایل ها را فقط به خود اپلیکیشن محدود نمایید (فایل های اپلیکیشن را `private` نگه دارید) و در صورت نیاز به اشتراک گذاری داده ها بین اپلیکیشن ها از کامپوننت نرم افزاری `content provider` استفاده کنید.

مثال زیر استفاده از API مورد نظر را به نمایش می گذارد.

```
public class Util {
    public static void writeConfiguration(Context ctx) {
        try (FileOutputStream openFileOutput =
            ctx.openFileOutput("config.txt", Context.MODE_PRIVATE);) {
            openFileOutput.write("This is a test1.".getBytes());
            openFileOutput.write("This is a test2.".getBytes());
        } catch (Exception e) {
            // not handled
        }
    }
    public void readFileFromInternalStorage(String fileName) {
        String eol = System.getProperty("line.separator");
        try (BufferedReader input = new BufferedReader(new InputStreamReader(
            openFileInput(fileName)))) {
            String line;
            StringBuffer buffer = new StringBuffer();
            while ((line = input.readLine()) != null) {
                buffer.append(line + eol);
            }
        } catch (Exception e) {
            // we do not care
        }
    }
}
```

## 4-9-2-حافظه و محل ذخیره سازی خارجی (external storage)

اندروید از حافظه خارجی مانند SD و دسترسی به آن نیز پشتیبانی می کند. تمامی فایل ها و پوشه های مستقر در حافظه خارجی برای کلیه اپلیکیشن هایی که دارای مجوز لازم هستند، قابل دسترسی می باشد.

جهت خواندن داده از حافظه خارجی، اپلیکیشن شما می بایست از مجوز android.permission.READ\_EXTERNAL\_STORAGE برخوردار باشد.

به منظور درج و نوشتن داده در حافظه خارجی، اپلیکیشن شما می بایست از مجوز android.permission.WRITE\_EXTERNAL\_STORAGE برخوردار باشد. برای دسترسی به حافظه خارجی، کافی است متد Environment.getExternalStorageDirectory() را فراخوانی نمایید.

با فراخوانی متد زیر شما می توانید وضعیت حافظه خارجی را بررسی کنید. چنانچه دستگاه اندروید از طریق USB به کامپیوتر متصل شده است، در آن صورت کارت SD که به عنوان حافظه خارجی مورد استفاده قرار گرفته است، از دسترس خارج می شود.

Environment.getExternalStorageState().equals(Environment.MEDIA\_MOUNTED)

تکه کد زیر مثالی از خواندن اطلاعات از حافظه خارجی را به نمایش می گذارد.

```
private void readFileFromSDCard() {
    File directory = Environment.getExternalStorageDirectory();
    // assumes that a file article.rss is available on the SD card
    File file = new File(directory + "/article.rss");
    if (!file.exists()) {
        throw new RuntimeException("File not found");
    }
    Log.e("Testing", "Starting to read");
    BufferedReader reader = null;
    try {
        reader = new BufferedReader(new FileReader(file));
        StringBuilder builder = new StringBuilder();
        String line;
        while ((line = reader.readLine()) != null) {
            builder.append(line);
        }
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        if (reader != null) {
```

```
        try {
            reader.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```



400

آدرس آموزشگاه : تهران - خیابان شریعتی - بالاتر از خیابان ملک - جنب بانک صادرات - پلاک 651 طبقه دوم - واحد 7

88146323 - 88446780 - 88146330

## بخش چهارم :

# اتصال به اینترنت، اجرای عملیات HTTP و دسترسی به منابع موجود در سطح وب در اندروید / Android networking

این آموزش نحوه‌ی دسترسی به منابع و محتوا از طریق HTTP را برای شما شرح می‌دهد.

پروژه‌ی آموزش حاضر در محیط برنامه نویسی Eclipse نوشته شده، مبتنی بر ویرایش 1.6 زبان Java و ورژن 5.0 سیستم عامل اندروید می‌باشد.

## 10-4-مروری بر اتصال به اینترنت و دسترسی به منابع از اینترنت در اندروید

### 10-4-1-دسترسی به اینترنت در اندروید

اندروید پکیج java.net را شامل می‌شود. با وارد کردن این پکیج در پروژه‌ی می‌توانید به منابع و محتوای مورد نیاز از طریق شبکه دسترسی پیدا کنید. کلاس پایه برای اتصال HTTP در پکیج HttpURLConnection، کلاس HttpURLConnection.java.net است.

اندروید حاوی اطلاعات استاندارد شبکه جاوا است.

می‌تواند از منابع دسترسی به شبکه موجود استفاده کند، به عنوان کلاس اصلی Net Package شبکه دسترسی HTTP در جاوا است، کلاس HttpURLConnection یک کلاس NetPackage است.

اجرای عملیات مربوط به شبکه با API جاوا طاقت فرسا است. بدین معنی که توسعه دهنده می‌باشد connection را باز کند/بندد، cache را فعال نموده و مطمئن شود که عملیات مربوط به شبکه در background thread (به صورت موازی و در پس زمینه) اجرا می‌شوند.

تعداد زیادی کتابخانه‌ی کد باز (open source) وجود دارد که در اختیار برنامه نویس قرار گرفته و انجام عملیات مذبور را آسان می‌سازد. پرکاربردترین این کتابخانه‌ها به شرح زیر می‌باشند:

- HTTP برای اتصال و دسترسی بهینه OkHttp – OkHttp efficient HTTP access •
- REST برای سرویس گیرنده های مبتنی بر Retrofit – Retrofit REST based clients •
- برای پردازش تصویر Glide – Glide image processing •

## 4-10-2-مجوز اتصال به اینترنت

به منظور اتصال به اینترنت، اپلیکیشن شما به مجوز android.permission.INTERNET احتیاج دارد. در ورژن های جدید کتابخانه های اندروید (API version API های نوین)، این مجوز به صورت خودکار به اپلیکیشن اعطا می شود.

## 4-10-3-بررسی وضعیت اتصال به اینترنت

دستگاه های اندروید همیشه به اینترنت دسترسی ندارند. برای بررسی اینکه آیا دستگاه مورد نظر به شبکه و اینترنت دسترسی دارد یا خیر، اپلیکیشن شما بایستی مجوز android.permission.ACCESS\_NETWORK\_STATE را داشته باشد.

جهت بررسی وضعیت اتصال به اینترنت، می توانید از کد زیر استفاده کنید.

```
public boolean isNetworkAvailable() {
    ConnectivityManager cm = (ConnectivityManager)
        getSystemService(Context.CONNECTIVITY_SERVICE);
    NetworkInfo networkInfo = cm.getActiveNetworkInfo();
    // if no network is available networkInfo will be null
    // otherwise check if we are connected
    if (networkInfo != null && networkInfo.isConnected()) {
        return true;
    }
    return false;
}
```

همان طور که در بالا گفته شد، برای بررسی وضعیت اتصال به شبکه، اپلیکیشن شما به مجوز ACCESS\_NETWORK\_STATE احتیاج دارد.

## 4-10-3-روش های بهینه برای اتصال و دسترسی به اینترنت در اندروید

در اپلیکیشن های اندرویدی، شما باید از اجرای عملیات طولانی و سنگین در UI thread خودداری نمایید. از جمله ای این عملیات می توان به اتصال به اینترنت و دسترسی به فایل اشاره کرد.

از ویرایش 3.0 به بعد اندروید، سیستم طوری تعبیه شده است که اگر دسترسی به اینترنت در UI thread رخ دهد، خطای NetworkOnMainThreadException صادر شده و به تبع آن سیستم به طور ناگهانی از کار می‌افتد.

روش معمول و بهینه برای اتصال به اینترنت و دسترسی به منابع سطح وب (در یک اپلیکیشن با کیفیت)، استفاده از یک سرویس است. گفتنی است که می‌توان از یک activity یا fragment نیز به اینترنت دسترسی را انجام داد، اما استفاده از سرویس برای نیل به این هدف طراحی بهینه تری را برای اپلیکیشن شما رقم می‌زند و همچنین به ساده نگه داشتن کد activity شما کمک شایانی می‌نماید.

**توجه:** جهت تست می‌توانید با درج تکه کد زیر در ابتدای متده `onCreate()` کلاس activity، دسترسی به اینترنت را در thread اصلی اپلیکیشن فراهم کنید.

```
StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder().permitAll().build();
StrictMode.setThreadPolicy(policy);
```

Unresolved directive in 001\_article.adoc - include::../JavaNetworking/010\_overview.adoc[] == Web Sockets

یک استاندارد مبتنی بر HTTP برای تبادل ناهمزمان پیغام (Web Socket asynchronous) بین سرویس گیرنده و سرویس دهنده است. به منظور راه اندازی این ارتباط، یک درخواست HTTP GET با یک HTTP header خاص ایجاد نمایید. اگر سرویس دهنده این درخواست را پذیرفت، سرویس گیرنده و سرویس دهنده می‌توانند با هم پیغام (داده هایی را) رد و بدل کنند.

این پیغام ها می‌توانند متن یا داده های binary باشند. لازم به ذکر است که داده های مورد تبادل می‌بایست کم حجم باشند چرا که پروتکل web socket اساساً به منظور انتقال payload و داده های کم حجم تعبیه و طراحی شده است.

بهترین روش برای تبادل پیغام و داده بین کلاینت و سرور، قرار دادن آن در قالب JSON می‌باشد.

پیام می توانند متنی و یا داده های باینری و نسبتا کوچک باشند ، به عنوان یک وب سوکت پروتکل در نظر گرفته شوندکه با محموله های کوچک در داده استفاده می شوند.

این تمرین خوبی به استفاده از JSON به عنوان فرمت داده ها را برای پیام است .



بخش پنجم :

## استفاده از کتابخانه `RestClient` به عنوان `Retrofit 2.0`

[Retrofit-4-11](#)

404

آدرس آموزشگاه : تهران - خیابان شریعتی - بالاتر از خیابان ملک - جنب بانک صادرات - پلاک 651 طبقه دوم - واحد 7

88146323 - 88446780 - 88146330

عبارت است از یک REST Client برای Android و Java که توسط Square که توسعه داده است ارائه می شود. این کتابخانه، بازیابی و بارگذاری JSON یا هر داده‌ی ساخت یافته‌ی دیگری را از طریق یک وب سرویس مبتنی بر REST انجام می‌دهد. Retrofit را می‌توان با یک converter تنظیم نموده و برای serialize داده‌ها مورد استفاده قرار داد. معمولاً برای داده‌هایی که در فرمت JSON ذخیره شده‌اند از Gson استفاده می‌شود، با این حال شما می‌توانید converter‌های اختصاصی و دلخواه خود را جهت پردازش XML یا دیگر پروتکل‌ها مورد استفاده قرار دهید.

از کتابخانه‌ی OkHttp برای مدیریت درخواست‌های HTTP بهره می‌گیرید.

به شما این امکان را می‌دهد از Converter‌های زیر استفاده نمایید.

- Gson: com.squareup.retrofit:converter-gson
- Jackson: com.squareup.retrofit:converter-jackson
- Moshi: com.squareup.retrofit:converter-moshi
- Protobuf: com.squareup.retrofit:converter-protobuf
- Wire: com.squareup.retrofit:converter-wire
- Simple XML: com.squareup.retrofit:converter-simplexml

برای کار با Retrofit به سه کلاس زیر احتیاج دارد.

1. کلاس model ویژه‌ی نگاشت داده‌های JSON.
2. Interface‌هایی که عملیات و توابع HTTP را تعریف می‌کنند.
3. کلاس Retrofit – نمونه‌ای که از این interface استفاده می‌کند. Builder API امکان تعريف آدرس URL (آدرس سرویس) (آدرس سرویس) endpoint را برای عملیات HTTP فراهم می‌آورد.

می‌توانید با مراجعه به سایت زیر، داده‌های مبتنی بر JSON خود را به فرمت POJO (آجکت‌های ساده و بدون متاداده) تبدیل نمایید.

## تمرین: استفاده از Retrofit برای کوئری گرفتن و پرس و جو از Stackoverflow در اندروید

### هدف از این تمرین

یک سایت پر بازدید است که برنامه نویس ها مشکلات برنامه نویسی خود را در آن مطرح می کنند. این سایت یک REST API ارائه می کند که به خوبی مستندسازی شده و با توضیحات لازم در اختیار توسعه دهنگان قرار می گیرد. Query ها را می توان با استفاده از این API ساخت. می توانید جهت مشاهده ای مستندات این API و استفاده از قابلیت های آن به آدرس <https://api.stackexchange.com/docs/search> مراجعه نمایید. در تمرین زیر با کتابخانه `retrofit` و بر اساس تگ هایی که مشخص می شود، می توانید از سوال های پاسخ داده یا تایید نهایی نشده کوئری بگیرید.

در مثال جاری URL کوئری زیر را بکار می بریم. تگ ها از طریق کد ما مشخص می شوند.

<https://api.stackexchange.com/2.2/search?order=desc&sort=activity&tagged=android&site=stack overflow>

### آموزشگاه تحلیلکردن اندروید

#### 4-11-4- ساخت پروژه و تنظیمات اولیه

یک اپلیکیشن اندروید به نام `com.vogella.android.retrofitstackoverflow` ایجاد نمایید. این اپلیکیشن باستثنی اسم `com.vogella.android.retrofitstackoverflow` را به عنوان top level package مورد استفاده قرار دهد.

کتابخانه `retrofit` را به فایل `build.gradle` اضافه نمایید.

```
compile 'com.squareup.retrofit:retrofit:2.0.0-beta2'
compile 'com.squareup.retrofit:converter-gson:2.0.0-beta2'
```

مجوز لازم برای دسترسی به اینترنت را در فایل تنظیمات (manifest) اپلیکیشن تنظیم نمایید.

## 4-11-2- Retrofit adapter و کلاس API

در پاسخ JSON ای که از StackOverflow دریافت می شود، تنها به title و link نیاز داریم. کلاس های زیر را ایجاد نمایید.

```
package android.vogella.com.retrofitstackoverflow;
// This is used to map the JSON keys to the object by GSON
public class Question {
    String title;
    String link;
    @Override
    public String toString() {
        return(title);
    }
}
```

```
package android.vogella.com.retrofitstackoverflow;
import java.util.List;
public class StackOverflowQuestions {
    List<Question> items;
}
```

با پیاده سازی REST API زیر Retrofit را برای interface تعریف نمایید.

**توجه:** از آنجایی stackoverflow پاسخ را در یک آبجکت جاسازی می کند، لازم است ساختار داده ای مورد نیاز را از نوع list تعریف نمایید.

```
package com.vogella.android.retrofitstackoverflow;
import retrofit.Callback;
import retrofit.http.GET;
import retrofit.http.Query;
import retrofit.Call;
public interface StackOverflowAPI {
    @GET("/2.2/questions?order=desc&sort=creation&site=stackoverflow")
    Call<StackOverflowQuestions> loadQuestions(@Query("tagged") String tags);
}
```

یک منوی XML مشابه زیر با نام main\_menu.xml ایجاد نمایید.

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/menu_load"
        android:title="Load" />
</menu>
```

کد کلاس activity خود را طوری ویرایش نمایید که به شما امکان کوئری گرفتن از سوال هایی را بدهد که با رشته‌ی "android" علامت یا تگ گذاری شده‌اند.

```
package com.vogella.android.retrofitstackoverflow;
import android.app.Activity;
import android.app.ListActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.Window;
import android.widget.ArrayAdapter;
import android.widget.Toast;
import java.util.ArrayList;
import retrofit.Call;
import retrofit.Callback;
import retrofit.GsonConverterFactory;
import retrofit.Response;
import retrofit.Retrofit;
public class MainActivity extends ListActivity implements Callback<StackOverflowQuestions> {
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_INDETERMINATE_PROGRESS);
        requestWindowFeature(Window.FEATURE_PROGRESS);
        ArrayAdapter<Question> arrayAdapter =
            new ArrayAdapter<Question>(this,
                android.R.layout.simple_list_item_1,
                android.R.id.text1,
                new ArrayList<Question>());
        setListAdapter(arrayAdapter);
        setProgressBarIndeterminateVisibility(true);
        setProgressBarVisibility(true);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main_menu, menu);
        return true;
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        setProgressBarIndeterminateVisibility(true);
        Retrofit retrofit = new Retrofit.Builder()
            .baseUrl("https://api.stackexchange.com")
```

```

.addConverterFactory(GsonConverterFactory.create())
.build();
// prepare call in Retrofit 2.0
StackOverflowAPI stackOverflowAPI = retrofit.create(StackOverflowAPI.class);
Call<StackOverflowQuestions> call = stackOverflowAPI.loadQuestions("android");
//asynchronous call
call.enqueue(this);
// synchronous call would be with execute, in this case you
// would have to perform this outside the main thread
// call.execute()
// to cancel a running request
// call.cancel();
// calls can only be used once but you can easily clone them
//Call<StackOverflowQuestions> c = call.clone();
//c.enqueue(this);
return true;
}
@Override
public void onResponse(Response<StackOverflowQuestions> response, Retrofit retrofit) {
    setProgressBarIndeterminateVisibility(false);
    ArrayAdapter<Question> adapter = (ArrayAdapter<Question>) getListAdapter();
    adapter.clear();
    adapter.addAll(response.body().items);
}
@Override
public void onFailure(Throwable t) {
    Toast.makeText(MainActivity.this, t.getLocalizedMessage(), Toast.LENGTH_SHORT).show();
}
}

```

## تمرین: استفاده از Retrofit برای دسترسی به توابع Github (API) در

اندروید

تنظیم پروژه

به منظور تست Retrofit، یک اپلیکیشن به نام Retrofit Github ایجاد نمایید. اسم  
برای این پروژه top level package را به عنوان com.vogella.android.retrofitgithub  
دهید.

برای استفاده از Retrofit در اپلیکیشن مورد نظر، کتابخانه (dependency) زیر را به فایل  
build.gradle خود اضافه نمایید.

```
compile 'com.squareup.retrofit:retrofit:2.0.0-beta2'
compile 'com.squareup.retrofit:converter-gson:2.0.0-beta2'
```

مجوز لازم برای دسترسی به اینترنت را در فایل تنظیمات اپلیکیشن (manifest) تنظیم نمایید.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.vogella.android.retrofitgithub" >
    <uses-permission android:name="android.permission.INTERNET"/>
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

و کلاس زیر را برای Retrofit API ایجاد نمایید.

```
package com.vogella.android.retrofitgithub;

// This is used to map the JSON keys to the object by GSON
public class GithubRepo {

    String name;
    String url;

    @Override
    public String toString() {
        return(name + " " + url);
    }
}

package com.vogella.android.retrofitgithub;

// This is used to map the JSON keys to the object by GSON
public class GithubUser {

    String login;
    String name;
    String email;

    @Override
    public String toString() {
        return(login);
    }
}
```

```

import java.util.List;

import retrofit2.Call;
import retrofit2.http.GET;
import retrofit2.http.Path;

public interface GithubAPI {
    String ENDPOINT = "https://api.github.com";

    @GET("/users/{user}")
    Call<GithubUser> getUser(@Path("user") String user);

    @GET("users/{user}/repos")
    Call<List<GithubRepo>> getRepos(@Path("user") String user);
}

package com.vogella.android.retrofitgithub;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;

import java.util.List;

import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;
import retrofit2.Retrofit;
import retrofit2.converter.gson.GsonConverterFactory;

public class MainActivity extends Activity implements Callback<GithubUser> {

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

    }
    public void onClick(View view) {
        Gson gson = new GsonBuilder()
            .setDateFormat("yyyy-MM-dd'T'HH:mm:ssZ")
            .create();
        Retrofit retrofit = new Retrofit.Builder()
            .baseUrl(GithubAPI.ENDPOINT)
            .addConverterFactory(GsonConverterFactory.create(gson))
            .build();
    }
}

```

```

GithubAPI githubUserAPI = retrofit.create(GithubAPI.class);

switch (view.getId()) {
    case R.id.loadUserData:
        // prepare call in Retrofit 2.0

        Call<GithubUser> callUser = githubUserAPI.getUser("vogella");
        //asynchronous call
        callUser.enqueue(this);
        break;
    case R.id.loadRepositories:
        Call<List<GithubRepo>> callRepos = githubUserAPI.getRepos("vogella");
        //asynchronous call
        callRepos.enqueue(repos);
        break;
}
}

Callback repos = new Callback<List<GithubRepo>>(){

    @Override
    public void onResponse(Call<List<GithubRepo>> call, Response<List<GithubRepo>> response) {
        if (response.isSuccessful()) {
            List<GithubRepo> repos = response.body();
            StringBuilder builder = new StringBuilder();
            for (GithubRepo repo: repos) {
                builder.append(repo.name + " " + repo.toString());
            }
            Toast.makeText(MainActivity.this, builder.toString(), Toast.LENGTH_SHORT).show();
        } else {
            Toast.makeText(MainActivity.this, "Error code " + response.code(), Toast.LENGTH_SHORT).show();
        }
    }

    @Override
    public void onFailure(Call<List<GithubRepo>> call, Throwable t) {
        Toast.makeText(MainActivity.this, "Did not work " + t.getMessage(), Toast.LENGTH_SHORT).show();
    };
}

@Override
public void onResponse(Call<GithubUser> call, Response<GithubUser> response) {
    int code = response.code();
    if (code == 200) {
        GithubUser user = response.body();
        Toast.makeText(this, "Got the user: " + user.email, Toast.LENGTH_LONG).show();
    } else {
        Toast.makeText(this, "Did not work: " + String.valueOf(code), Toast.LENGTH_LONG).show();
    }
}
}

```

```

        }
    }

@Override
public void onFailure(Call<GithubUser> call, Throwable t) {
    Toast.makeText(this, "Nope", Toast.LENGTH_LONG).show();
}

}

```

url مقابله با <https://api.github.com/users/vogella> را در نوار آدرس مرورگر وارد نمایید. پس از پیمایش به آدرس مذکور، یک کلاس data model و interface با داده های مورد نظر ایجاد کنید. حال این داده ها را با Retrofit خوانده و در یک لیست به نمایش بگذارد.



بخش ششم :

## آموزش RxJava 2.0

### RxJava 2.0-12-4

#### RxJava-4-12-1 و شرح مفهوم برنامه نویسی Reactive یا ناهمzman

هر برنامه ای که دارای چهار ویژگی یا پارامتر (قابلیت مدیریت خطاهای و ارائه ای بهترین سرویس)، (انعطاف و مقایس پذیری) scalability، (واکنش گرا و تعاملی) responsive و (رویداد reactive) Event-driven باشد را در اصطلاح reactive می نامند. در مدل برنامه نویسی reactive محور سرویس گیرنده یا کاربر به محض ورود داده به آن واکنش نشان می دهند. به همین خاطر برنامه نویسی asynch را برنامه نویسی reactive نیز می خوانند. مدل برنامه نویسی reactive این امکان را فراهم می آورد تا تغییرات event ها به observer ها (توابع) گوش فرا دهنده و ثبت شده منتشر (propagate) و اعمال نمایید.

در واقع RxJava امکان پیاده سازی وظایف غیرهمزمان را به راحتی برای توسعه دهنده فراهم می آورد.

RxJava پیاده سازی از ReactiveX برای Java می باشد که NetFlix برای توسعه داده و در بین برنامه نویسان جاوا محبوبیت زیادی پیدا کرده است. RxJava در سال 2014 کد باز (open source) شده و تحت آدرس <http://reactivex.io/> میزبانی می شود. ورژن جاوایی آن تحت قابل دسترسی می باشد. این کتابخانه تحت لیسانس <https://github.com/ReactiveX/RxJava> Apache 2.0 منتشر می شود.

RxJava خود را تحت عنوان یک API که برای برنامه نویسی ناهمزمان طراحی شده (تعریف تسلیک ها و وظایف ناهمزمان) و از آبجکت های observable stream اطلاعاتی را منتشر کرده و آن ها را به توابع گوش فرادهنده به تغییرات که همان observer ها هستند ارسال می کنند) استفاده می کند، معرفی می نماید.

## 12-4- اضافه کردن کتابخانه RxJava 2.0

از زمان نگارش این مقاله، ویرایش جدیدتری از کتابخانه RxJava ارائه شده است. مقدار `g.a.y` را با `2.0.1` یا نسخه‌ی جدیدتر جایگزین نمایید.

برای افزودن کتابخانه‌ی مزبور ویژه‌ی سیستم کامپایل Maven، می‌توانید تکه کد زیر را در بخش dependency اضافه کتابخانه اضافه نمایید.

```
<dependency>
  <groupId>io.reactivex.rxjava2</groupId>
  <artifactId>rxjava</artifactId>
  <version>g.a.v</version>
</dependency>
```

برای سیستم کامپایل Gradle، می‌توانید RxJava را از طریق دستور زیر اضافه نمایید.

```
compile group: 'io.reactivex.rxjava2', name: 'rxjava', version: 'g.a.v'
```

### 4-12-3- برنامه نویسی ناهزمان (Async)

امروزه برنامه نویسی به سبک دستوری/imperative طوری که اپلیکیشن در لحظه تنها یک عملیات را پردازش کند (single-thread) دیگر به هیچ وجه کارامد نیست و ممکن است UI های unresponsive را مسدود کند و در نهایت تجربه کاربری ضعیفی را در پی داشته باشد.

با مدیریت رخدادهای پیش بینی نشده به صورت ناهمزمان (async)، می توان از این اتفاق جلوگیری کرد. برای مثال، چنانچه لازم باشد منتظر فراخوانی وب سرویس یا کوئری دیتابیس باشید، اگر شبکه پاسخگو نبود (responsive)، قطعاً اپلیکیشن هنگ خواهد کرد.

مثالی در این زمینه:

```
public List<Todo> getTodos() {
    List<Todo> todosFromWeb = // query a webservice (with bad network latency)
    return todosFromDb;
}
```

فراخوانی متدهای getTodos() از thread اصلی یا یک UI thread سبب می شود تا زمان رسیدن todosFromWeb بوده و تجربه کاربری ضعیفی برای کاربر رقم خورده شود.

برای بهبود کارایی این query که مدت زمان اجرای آن مشخص نیست، می بایست آن را در یک thread دیگر اجرا نموده و سپس به هنگام رسیدن نتیجه، thread اصلی را مطلع نمایید.

```
public void getTodos(Consumer<List<Todo>> todosConsumer) {
    Thread thread = new Thread(()-> {
        List<Todo> todosFromWeb = // query a webservice
        todosConsumer.accept(todosFromWeb);
    });
    thread.start();
}
```

حال پس از فراخوانی متدهای getTodos(Consumer<List<Todo>> todosConsumer)، نخواهد می تواند بدون متوقف شدن همچنان به اجرا ادامه دهد و زمانی که متدهای accept از consumer صدا خورده شد، واکنش نشان دهد.

## 4-12-4 Subscription ها و Observable ها

RxJava از مفهوم Observable ها و Observer ها استفاده می کند. Observable ها توابعی هستند که با گوش فراخوانی و subscribe کردن به Observable ها که آبجکت هستند، از تغییرات رخداده بلافاصله آگاه شده و آن تغییرات را در خود منعکس می نمایند. Observer ها به مجرد اینکه یک observable مقداری را emit (ارسال) می کند، از آن مطلع می شوند. همچنین زمانی که observable اطلاعاتی را مبنی بر اینکه دیگر مقادیری وجود ندارد ارسال می کند، observer ها از آن مطلع می شوند. زمانی هم که observable با خطایی مواجه می شود، متدهای onCompleted() و onError(Throwable e) از interface Observer هستند که همگی از آن آگاه می شود. توابع مربوطه ارث بری می شوند. آبجکت یا نمونه ای از Subscription بیانگر اتصال و ارتباط بین یک observer و observable است. اگر متدهای unsubscribe() و dispose() از observable را بر روی این نمونه صدا بزنید، عضویت متدهای اتصال بین آنها قطع می شود. این متدهای observable می توانند زمانی مفید باشد که شما می خواهید بروز رسانی و آپدیت یک المان رابط کاربری یا widget خاتمه یافته (dispose) و از حافظه حذف شود.

## 4-12-5 thread برای گوش دادن به تغییرات

(observe)

متدهای observable در آن قرار است اجرا شود را با فراخوانی متدهای subscribeOn() و observeOn() تعريف نمایید و thread ای که observer ها در آن اجرا می شوند را با اجرای متدهای observeOn() اعلان کنید.

## 13-4 Operator ها

می توانید عملیاتی را بر روی observer های خود ثبت کنید که به شما اجازه می دهد نتیجه و یک observable را پیش از ارسال آن به observer ویرایش و دستکاری نمایید. متند map به شما این امکان را می دهد یک Func1 ثبت کنید که ورودی را ترجمه و تبدیل می کند.

## 14-4-استفاده از delay و به تعویق اندختن تولید و ارسال نتیجه

با فراخوانی متند debounce(delay, TimeUnit.MILLISECONDS) بر روی observer، می توانید اعلان کنید که تنها زمانی تغییرات را ارسال یا به عنوان نتیجه ارسال (emit) نماید که مقدار مورد نظر پس از گذشت مدت زمان از پیش تعیین شده تغییر نکرده باشد.

### 14-4-ایجاد Observable ها و Observer ها

RxJava متدهای زیادی برای ایجاد observable در اختیار توسعه دهنده قرار می دهد. این متدها عبارت اند از:

- یک Observable به عنوان ظرف ایجاد می کند که انواع داده ای را دربرمی گیرد. به عبارت دیگر این متند یک آبجکت را به observable ارسال می کند یا یک یا چند object را به یک Observable تبدیل می کند و سپس آن آبجکت یا آبجکت ها را به observer می فرستد.
- یک collection یا آرایه گرفته، مقادیر آن ها را به ترتیب موجود در ساختار داده ای مربوطه، emit یا به عنوان نتیجه ارسال می کند. در واقع این متند یک آرایه دریافت کرده، به ازای هر المان در ساختار داده ای مزبور، تابع observer را صدا می زند و مقدار را می فرستد.
- به شما امکان ساخت یک observable برای Callable را Observable.fromCallable() می دهد.

به منظور ساخت observable ها:

- Action1 را پیاده سازی نمایید - به شما این امکان را می دهد تا observer ساده ایجاد کنید که یک متده به نام call دارد. در صورت تولید یا ارسال (emit) آبجکت جدید، این متده صدا خورده می شود.

## مثالی از فراخوانی متده Observable.just()

تابع Observable یک observable ایجاد می کند و زمانی که یک observer به این گوش می دهد یا برای آن onNext() متد subscribe می کند، متد onNext() از observable ورودی observable.just() فراخوانی می شود.

```
import java.util.Arrays;
import java.util.List;
import rx.Observable;
public class RxJavaExample {
    public static void main(String[] args) {
        List<String> list = Arrays.asList("Android", "Ubuntu", "Mac OS"); 1
        Observable<List<String>> listObservable = Observable.just(list); 2
        listObservable.subscribe(new Observer<List<String>>() {
            @Override
            public void onCompleted() {}
            @Override
            public void onError(Throwable e) {}
            @Override
            public void onNext(List<String> list) {
                System.out.println(list);
            }
        });
    }
}
```

1. یک لیست ایجاد می کند.

2. observable را ایجاد می کند.

3. observable را برای گوش دادن به ثبت می کند.

## 4-15- انجام عملیات تبدیل

در زیر مثال ساده ای از RxJava را مشاهده می کنید که نوعی عملیات تبدیل در آن رخ داده است.

```
import java.util.Arrays;
import java.util.List;
import rx.Observable;
public class RxJavaExample {
    public static void main(String[] args) {
        List<String> list = Arrays.asList("Hello", "Streams", "Not");
        Observable.from(list)
            .filter(s -> s.contains("e"))
            .map(s -> s.toUpperCase())
            .reduce(new StringBuilder(), StringBuilder::append)
            .subscribe(System.out::print, e -> {},
                      () -> System.out.println("!"));
    }
}
```

## Subject\_4-14-2

Subject ها در واقع یک نوع پل ارتباطی یا پیشکار (proxy) هستند که خود هر دو نقش Observable و Observer را یکجا ایفا می کنند. Subject ها را می توان به عنوان یک pipe یا پل ارتباطی که داده ها را تبدیل و ترجمه می کنند در نظر گرفته و مورد استفاده قرار داد. به عنوان مثال می توان به PublishSubject اشاره کرد. همین که داده ای به PublishSubject ارسال می شود، بلافاصله این داده به بیرون ارسال می گردد. به عبارت دیگر Subject یک آجکت در RxJava هست که از تمامی ویژگی ها و property های Observable و Observer برخوردار Subscriber/Observer می باشد. هم می تواند به Observable ها گوش داده و از آن ها داده تحويل بگیرد و هم قادر است داده ها را به Observer هایی که به آن گوش می دهند، داده به عنوان خروجی ارسال کند. همچنین می تواند داده هایی را با فراخوانی مستقیم متده onNext() ارسال کند. از این طریق، Subject می تواند نقش واسط را ایفا کند که داده ها را دریافت کرده، عملیاتی را بروی آن انجام می دهد و سپس آن را به گوش فراخوان یا subscriber های خود ارسال می کند.

## 4-15-آجکت Promise یا همان Single (مکان نگهدار مقدار مورد نظر)

مدل برنامه نویسی reactive برای برقراری ارتباط دو طرفه و واکنشی از دو مفهوم Observer ها و Observable ها استفاده می کند.

امروزه استفاده از promise ها در فراخوانی های ناهمزمان (async) به ویژه همراه با زبان سمت Promise سرویس گیرنده‌ی JavaScript، از محبوبیت ویژه و کاربرد فراوانی برخودار شده است. در اصل جانگهدار یا placeholder برای یک مقدار دلخواه است که observer ها یا متدهای گوش فرا دهنده‌ی خود را به مجرد دریافت مقدار مورد انتظار و به اصطلاح موعود با خبر می‌سازد. RxJava نیز آبجکت یا نوعی به نام Single دارد که از لحاظ کاربرد بسیار شبیه به Promise است. نمونه یا آبجکت‌هایی که از کلاس Single ساخته می‌شود در واقع وظیفه‌ای مشابه وظیفه‌ی آبجکت Observable ایفا می‌نماید، با این تفاوت که دو متده بازفرایوان بیشتر به نام‌های onError() و onSuccess() ندارد.

```
import io.reactivex.Single;
public Single<List<Todo>> getTodos() {
    return Single.create(emitter -> {
        Thread thread = new Thread(() -> {
            try {
                List<Todo> todosFromWeb = // query a webservice
                emitter.onSuccess(todosFromWeb);
            } catch (Exception e) {
                emitter.onError(e);
            }
        });
        thread.start();
    });
}
```

نحوه‌ی استفاده از نمونه‌ی کلاس Single در قطعه کد زیر به نمایش گذاشته شده است.

```
import io.reactivex.Single;
import io.reactivex.disposables.Disposable;
import io.reactivex.observers.DisposableSingleObserver;
Single<List<Todo>> todosSingle = getTodos();
todosSingle.subscribeWith(new DisposableSingleObserver<List<Todo>>() {
    @Override
    public void onSuccess(List<Todo> todos) {
        // work with the resulting todos
    }
    @Override
    public void onError(Throwable e) {
        // handle the error case
    }
});
```

## 16-دور انداختن subscription ها (قطع اتصال بین observable و observer) و استفاده از کلاس CompositeDisposable

گوش فراخوان ها یا subscriber هایی که به آبجکت مربوطه متصل بوده و از تغییرات آن آگاه می شوند، طبیعتاً قرار نیست تا ابد به همین کار ادامه دهند. برای مثال هم ممکن است به خاطر تغییر کوچکی در وضعیت، event ارسالی از کلاس Observable دیگر مورد نیاز و درخور توجه نباشد.



```

import io.reactivex.Single;
import io.reactivex.disposables.Disposable;
import io.reactivex.observers.DisposableSingleObserver;
Single<List<Todo>> todosSingle = getTodos();
Disposable disposable = todosSingle.subscribeWith(new DisposableSingleObserver<List<Todo>>() {
    @Override
    public void onSuccess(List<Todo> todos) {
        // work with the resulting todos
    }
    @Override
    public void onError(Throwable e) {
        // handle the error case
    }
});
// continue working and dispose when value of the Single is not interesting any more
disposable.dispose();

```

**توجه:** کلاس Single و دیگر کلاس های observable توابع متعددی جهت subscribe و گوش فراخوانی به تغییرات ارائه می دهند که همگی در خروجی یک آبجکت Disposable بازگردانی می نمایند.

به هنگام کار با چندین subscription (نشانگر ارتباط دو طرفه بین observer و observable) که ممکن است به دلیل تغییر در وضعیت، رخداد ارسالی از آن ها دیگر لازم و جالب توجه نباشد، می توان با بهره گیری از کلاس CompositeDisposable همزمان چندین subscription را دور انداخته و به اصطلاح از حافظه پاک نمود (dispose).

```

import io.reactivex.Single;
import io.reactivex.disposables.Disposable;
import io.reactivex.observers.DisposableSingleObserver;
import io.reactivex.disposables.CompositeDisposable;
CompositeDisposable compositeDisposable = new CompositeDisposable();
Single<List<Todo>> todosSingle = getTodos();
Single<Happiness> happiness = getHappiness();
compositeDisposable.add(todosSingle.subscribeWith(new DisposableSingleObserver<List<Todo>>() {
    @Override
    public void onSuccess(List<Todo> todos) {
        // work with the resulting todos
    }
    @Override
    public void onError(Throwable e) {
        // handle the error case
    }
})));
compositeDisposable.add(happiness.subscribeWith(new DisposableSingleObserver<List<Todo>>() {
    @Override
    public void onSuccess(Happiness happiness) {
        // celebrate the happiness :-D
    }
    @Override
    public void onError(Throwable e) {
        System.err.println("Don't worry, be happy! :-P");
    }
})));
// continue working and dispose all subscriptions when the values from the Single objects are not interesting
// any more
compositeDisposable.dispose();

```

## 17- ذخیره ی موقتی مقدار Observable های ارسال شده (completed observables)

هنگام کار با چند Observable و فراخوانی های ناهمزمان تمامی subscription های یک observable هایی که برای گوش دادن به آن ثبت شده اند) ضروری نبوده و منابع زیادی را مصرف می کند.

Observable ها معمولاً به بخش های مختلف اپلیکیشن پاس داده می شوند و هر بار لازم نیست با اضافه شدن یک subscription (و گوش دادن یک observable به observer) این فراخوانی سنگین اتفاق بیافتد.

کد زیر چهار با از یک وب سرویس کوئری می گیرد، هر چند یک بار کوئری گرفتن هم کفایت می کند.  
در واقع همان آبجکت های Todo نمایش داده می شوند اما هر بار به روش های مختلف.

```
Single<List<Todo>> todosSingle = Single.create(emitter -> {
    Thread thread = new Thread(() -> {
        try {
            List<Todo> todosFromWeb = // query a webservice
            System.out.println("Called 4 times!");
            emitter.onSuccess(todosFromWeb);
        } catch (Exception e) {
            emitter.onError(e);
        }
    });
    thread.start();
});
todosSingle.subscribe(... " Show todos times in a bar chart " ...);
showTodosInATable(todosSingle);
todosSingle.subscribe(... " Show todos in gant diagram " ...);
anotherMethodThatsSupposedToSubscribeTheSameSingle(todosSingle);
```

تکه کد بعدی از متده استفاده می کند، به همین خاطر آبجکت Single، پس از اینکه ارسال برای بار اول با موفقیت انجام شد، در حافظه ی موقت ذخیره می شود و از وب سرویس تنها یک بار کوئری گرفته می شود.

```
Single<List<Todo>> todosSingle = Single.create(emitter -> {
    Thread thread = new Thread(() -> {
        try {
            List<Todo> todosFromWeb = // query a webservice
            System.out.println("I am only called once!");
            emitter.onSuccess(todosFromWeb);
        } catch (Exception e) {
            emitter.onError(e);
        }
    });
    thread.start();
});
// cache the result of the single, so that the web query is only done once
Single<List<Todo>> cachedSingle = todosSingle.cache();
cachedSingle.subscribe(... " Show todos times in a bar chart " ...);
showTodosInATable(cachedSingle);
cachedSingle.subscribe(... " Show todos in gant diagram " ...);
anotherMethodThatsSupposedToSubscribeTheSameSingle(cachedSingle);
```

## Backpressure و Flowable<T>-18-4

در RxJava ممکن است به کرات با شرایطی مواجه شوید که در آن تعداد آیتم های ارسالی از یک Observable بیشتر از میزانی است که یک Observer قادر به استفاده از آن هست. در چنین شرایطی این مسئله مطرح می شود که چگونه می توان این انباشت آیتم های استفاده نشده را مدیریت نمود. به این رخداد در اصطلاح برنامه نویسی back pressure گفته می شود.

RxJava 2.0 یک نوع جدید به نام `Flowable<T>` را معرفی کرد که از لحاظ عملکرد بسیار شبیه به `Observable<T>` باشد با این تفاوت که `Flowable<T>` قابلیت مدیریت `back pressure` دارد و `Observable<T>` همچنین امکانی را فراهم نمی کند.

در 1.0 RxJava این مفهوم کمی دیر به نوع `Observable<T>` اضافه شد. اما یک مشکل وجود داشت: برخی خطای زمان اجرای `MissingBackpressureException` را می دادند. به همین خاطر `Flowable<T>` اضافه شده و این مشکل را برطرف نمود.

جدا از نوع `Completable<T>`, نوع `Observable<T>`, `Single<T>`, `Maybe<T>` و `Flowable<T>` نیز `back pressure` را اصلا پشتیبانی نمی کنند.

## آموزشگاه تحلیلکردن داده ها

## 19-4-تبدیل نوعی به نوع دیگر

می توان به راحتی یک نوع از RxJava را به نوع دیگر تبدیل نمود.

**Table 1.**  
Conversion  
between  
types

From / To	Flowable	Observable	Maybe	Single	Completable
Flowable		toObservable()	reduce() elementAt() firstElement() lastElement() singleElement()	scan() elementAt() first()/firstOrError() last()/lastOrError() single()/singleOrDefault() all()/any()/count() (and more...)	ignoreElements()
Observable	toFlowable()		reduce() elementAt() firstElement() lastElement() singleElement()	scan() elementAt() first()/firstOrError() last()/lastOrError() single()/singleOrDefault() all()/any()/count() (and more...)	ignoreElements()
Maybe	toFlowable()	toObservable()		toSingle() sequenceEqual()	toCompletable()
Single	toFlowable()	toObservable()	toMaybe()		toCompletable()
Completable	toFlowable()	toObservable()	toMaybe()	toSingle() toSingleDefault()	

## RxJava های Subscription و Observable - 20-4- تست

### 4-1- observable ها

کتابخانه‌ی RxJava یک کلاس به نام TestSubscriber در اختیار توسعه دهنده قرار می‌دهد که با استفاده از آن می‌توان یک observable را تست نمایید.

```
Observable<String> obs = ...// assume creation code here  
TestSubscriber<String> testSubscriber = new TestSubscriber<>();  
obs.subscribe(testSubscriber);  
testSubscriber.assertNoErrors();  
List<String> chickens = testSubscriber.getOnNextEvents();  
// TODO assert your string integrity...
```

مکانیزمی را در اختیار شما قرار می‌دهد که به واسطه‌ی آن می‌توانید scheduler های ارائه (exposed) شده را طوری بازنویسی نمایید که به صورت همزمان (sync) فراخوانی شوند. برای مثالی در این زمینه می‌توانید <http://fedepaol.github.io/blog/2015/09/13/testing-rxjava-observables-subscriptions/> مراجعه نمایید.



بخش هفتم :

## آموزش پیاده سازی پردازش پس زمینه ای و ناهمگام با Handler ، Loader و AsyncTask ،

این آموزش به شرح مفاهیم Thread ها و استفاده از کلاس AsyncTask می پردازد. همچنین مفهوم پردازش ناهمگام در اپلیکیشن های اندرویدی را تشریح می نماید. سپس نحوهی مدیریت چرخه ای حیات/lifecycle با thread ها و پردازش پس زمینه ای را تحت پوشش قرار می دهد.

پروژه های این مبحث در محیط برنامه نویسی Android Studio نوشته و تست می شوند.

### 21-4-پردازش پس زمینه ای در اندروید

#### 1-21-4-چرا استفاده از مفهوم همروندي/concurrency توصيه می شود؟

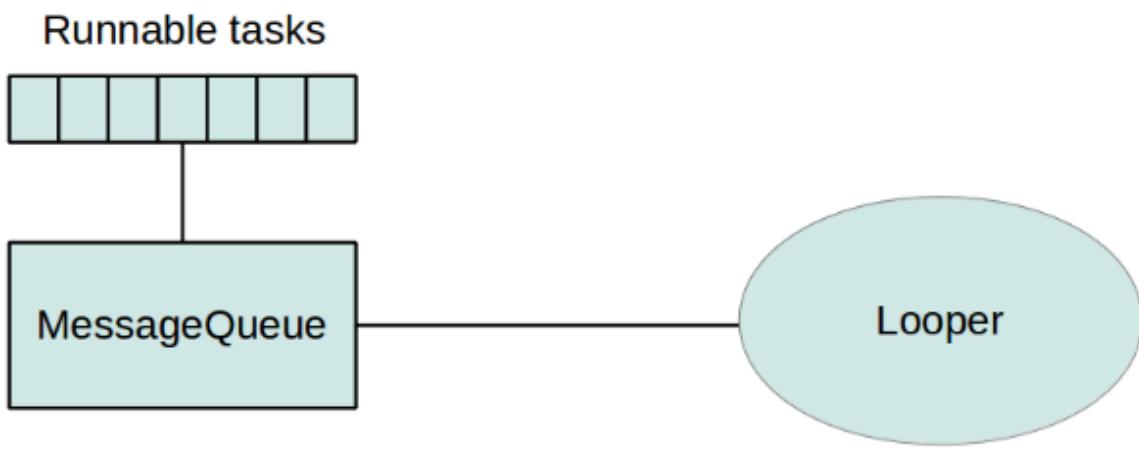
در حالت پیش فرض، یک اپلیکیشن اندرویدی در thread اصلی اجرا می شود. از اینرو تمامی دستورات به صورت پشت سرهم (خط به خط) و طبق یک ترتیب خاص اجرا می شوند. در چنین وضعیتی، اگر اپلیکیشن یک فرایند طولانی را راه اندازی کند، اپلیکیشن تا زمانی که عملیات مزبور به اتمام نرسیده، (thread اصلی مسدود شده) قادر به پردازش درخواست دیگری نخواهد بود.

به منظور ارائه ای تجربه ای کاربری بهینه، تمامی عملیات کند و طولانی می بايست در اپلیکیشن اندرویدی به صورت ناهمزمان (async) اجرا شوند. برای نیل به این هدف می توانید از ساختارهای مدیریت همروندي (concurrency) جawa یا چارچوب نرم افزاری (framework) اندروید استفاده نمایید. از عملیات کند می توان به پردازش های مربوط به اینترنت، فایل و دسترسی به داده از دیتابیس و محاسبات پیچیده اشاره کرد.

**نکته:** در اندروید چنانچه یک activity قادر نباشد با گذشت مدت 5 ثانیه از درخواست کاربر، پاسخ مناسب را ارائه دهد، سیستم بلاfaciale یک پنجره محاوره‌ی حاوی پیغام Application not responding (ANR) را به نمایش می‌گذارد. در این پنجره کاربر می‌تواند با انتخاب گزینه‌ی مربوطه، اپلیکیشن را کاملاً متوقف کند.

## UI thread 4-21-2

اندروید تمامی تسك‌ها و رخدادهای ورودی را در یک thread واحد به نام UI thread یا main thread مدیریت می‌نماید. در واقع سیستم اندروید تمامی رخدادها را در این thread، داخل یک صف جمع کرده و سپس event‌های قرار گرفته در این صف را با کمک نمونه‌ای از کلاس Looper پردازش می‌نماید. لازم به ذکر است که thread اصلی نمی‌تواند چندین عملیات را اداره کرده و در لحظه می‌تواند تنها یک رخداد ورودی را پردازش کند.



### 3-4-21-3-پردازش ناهمزمان و استفاده از thread ها در اندروید

اندروید برای پردازش `async` و ناهمزمان از کلاس `Thread` و جهت پردازش همزمان چند عملیات در پس زمینه پکیج `java.util.concurrent` را در اختیار توسعه دهنده قرار می دهد. این پکیج کلاس های `Executor` و `ThreadPools` را برای پردازش پس زمینه فراهم می نماید.

در صورت نیاز به آپدیت UI از یک `Thread` دیگر، لازم است با `main thread` همزمانی انجام دهید. به خاطر وجود این محدودیت ها، توسعه دهندگان اندروید اغلب از کدها و ساختارهای اختصاصی خود اندروید بهره می گیرند.

اندروید تعدادی ساختار ویژه جهت مدیریت همروندي (انجام چند پردازش در پس زمینه) ارائه می دهد که جدا از کدهای جاوا است.

از ساختارهای اختصاصی خود اندروید می توان از کلاس های `Handler` یا `android.os.AsyncTask` نام برد. روش های پیچیده تری ویژه ای مدیریت همروندي وجود دارد که لازمه ای آن، استفاده از کلاس `Loader`، `retained fragment` ها و سرویس ها است.

**توجه:** تا حد امکان از بکار بردن پنجره `ProgressBar` یا روش های مشابه که تعامل با UI را تا اتمام فرایند معینی مسدود می کند، خودداری نمایید. توصیه می شود برای این منظور یک توضیح مختصر UI همچنان قابلیت تعامل با کاربر را داشته باشد.

### 3-4-21-4-ارائه ای feedback و توضیح مختصر برای کاربر در طول اجرای عملیات طولانی

در طول اجرای عملیات طولانی، بهتر است توضیح مختصری درخصوص این عملیات برای کاربر به نمایش بگذارید.

می توانید این توضیحات را در قالب `action view` یا `action bar` در UI به نمایش بگذارید. روش دیگر این است که یک `ProgressBar` در `layout` تعریف کرده، آن را بر روی `visible` تنظیم نمایید و سپس در طول اجرای عملیات بروز رسانی کنید. هرچند توصیه می شود از روش اول استفاده

نمایید چرا که در آن صورت ارتباط کاربر با UI قطع نمی شود و اپلیکیشن همچنان responsive می ماند.

## Handler-22-4-کلاس

### 4-استفاده از کلاس Handler جهت مدیریت همروندي

ابتدا جهت پیاده سازی مفهوم همروندي در موبایل و ایجاد یک thread دیگر سوی thread اصلی، می بایست یک آبجکت از کلاس Thread ایجاد نمایید. سپس می توانید با کمک کلاس های handler و Message خروجی thread پس زمینه را به اصلی اطلاع دهید.

آبجکت Handler خود را داخل thread ای که در آن ایجاد می شود، ثبت می نماید. این آبجکت یک پل ارتباطی ایجاد کرده و داده ها از طریق آن به thread اصلی تحویل می دهد. به طور مثال، اگر شما یک نمونه ی جدید از Handler در متده است onCreate() کلاس activity خود ایجاد نمایید، در آن صورت می توانید از آبجکت ذکر شده به راحتی داده های مورد نیاز را به thread اصلی ارسال نمایید. داده هایی که کلاس Handler ارسال می کند، می توانند آبجکتی از کلاس Message یا Runnable باشد.

Handler به ویژه در شرایطی کارامد تلقی می شود که لازم باشد داده هایی را چند بار به thread اصلی ارسال (post) نمایید.

### 4-ایجاد و استفاده ی مجدد از thread اصلی

به منظور پیاده سازی کلاس handler، می بایست یک کلاس فرزند (subclass) از آن ایجاد کرده و متده handleMessage() آن را جهت پردازش پیغام ها بازنویسی (override) نمایید. سپس می توانید به راحتی با فراخوانی توابع sendMessage(Message) یا sendEmptyMessage() پیغام های مورد نیاز را به آن ارسال کنید. جهت ارسال یک آبجکت Runnable به کلاس مذکور، لازم است متده post() را بکار ببرید.

به منظور جلوگیری از ایجاد غیر ضروری آبجکت و اشغال حافظه، می توانید آبجکت Handler جاری کلاس activity خود را مجدد استفاده یا به اصطلاح بازیافت نمایید.

```
// Reuse existing handler if you don't
// have to override the message processing
handler = getWindow().getDecorView().getHandler();
```

کلاس View به شما این امکان را می دهد تا آبجکت هایی از جنس Runnable را به وسیله ی متده ارسال (post) نمایید (این متده به شما امکان می دهد تا پیغام پایان کار را به اصلی thread() اعلان نمایید).

## مثال

کد زیر نحوه ی استفاده از یک handler از view را نمایش می دهد. فرض کنید activity شما از فایل layout زیر جهت تنظیم ظاهر خود استفاده می کند.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <ProgressBar
        android:id="@+id/progressBar1"
        style="?android:attr/progressBarStyleHorizontal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:indeterminate="false"
        android:max="10"
        android:padding="4dip" >
    </ProgressBar>
    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="" >
    </TextView>
    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="startProgress"
        android:text="Start Progress" >
```

```
</Button>
</LinearLayout>
```

با توجه به کد activity زیر که برای برنامه نوشته شده، زمانی که کاربر بر روی دکمه یا آبجکت کلیک می کند، آبجکت ProgressBar بلافاصله بروز رسانی می شود.

```
package de.vogella.android.handler;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.ProgressBar;
import android.widget.TextView;
public class ProgressTestActivity extends Activity {
    private ProgressBar progress;
    private TextView text;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        progress = (ProgressBar) findViewById(R.id.progressBar1);
        text = (TextView) findViewById(R.id.textView1);
    }
    public void startProgress(View view) {
        // do something long
        Runnable runnable = new Runnable() {
            @Override
            public void run() {
                for (int i = 0; i <= 10; i++) {
                    final int value = i;
                    doFakeWork();
                    progress.post(new Runnable() {
                        @Override
                        public void run() {
                            text.setText("Updating");
                            progress.setProgress(value);
                        }
                    });
                }
            }
        };
        new Thread(runnable).start();
    }
    // Simulating something timeconsuming
    private void doFakeWork() {
        try {
            Thread.sleep(2000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

}

# AsyncTask کلاس-23-4

## 4-23-1- هدف استفاده از کلاس AsyncTask

کلاس AsyncTask به شما اجازه می دهد تا دستوراتی را در پس زمینه به اجرا بگذارید و سپس آن را با thread اصلی همگام نموده و همچنین گزارشاتی مربوط به ترسک های در حال اجرا به اغلب برای اجرای عملیات thread اصلی تحويل دهید. لازم به ذکر است که از کلاس AsyncTasks در پس زمینه که آن را بروز رسانی می کنند، استفاده می شود.

در واقع کلاس AsyncTask یک کلاس انتزاعی و abstract است که به برنامه های اندرویدی امکان می دهد تا thread اصلی (UI) را به صورت بهینه مدیریت نمایید. AsyncTask به توسعه دهنده کمک می کند تا تاسک هایی که چند ثانیه بیشتر طول نمی کشند را در پس زمینه به طور ناهمزمان اجرا نموده و نتیجه ی آن را در thread اصلی نمایش دهد.

#### ۲-۲۳-۴- استفاده از کلاس AsyncTask

برای استفاده از کلاس AsyncTask لازم است از آن یک کلاس فرزند ایجاد نمایید (از AsyncTask یک کلاس مشتق نمایید). کلاس AsyncTask از generic و varargs استفاده می کند. پارامترهای ورودی عبارتند از :

کلاس AsyncTask با صدا خوردن متد execute() راه اندازی می شود. متد execute() خود به تبع متدهای doInBackground() و onPostExecute() را فراخوانی می کند.

متد doInBackground() به عنوان آرگومان ورودی به تابع TypeOfVarArgParams فرستاده می شود. ویژه ای اطلاعات پیشرفت فرایند بکار برده می شود و ProgressValue خروجی از متد doInBackground() بازگردانی شود. این آرگومان خود به عنوان پارامتر ورودی به تابع onPostExecute() پاس داده می شود.

متد doInBackground() دربردارنده ای دستورات و کدهایی است که می بایست در یک thread مجزا در پس زمینه اجرا شوند.

متد onPostExecute() خود را با Thread اصلی (UI) همگام ساخته و بروز رسانی اطلاعات آن را به دنبال دارد. زمانی که عملیات متد doInBackground() به پایان می رسد، خود چارچوب نرم افزاری اندروید (FRAMEWORK) این متد را فراخوانی می کند.

### 23-3-4- اجرای همزمان چندین AsyncTasks

سیستم اندروید قبل از ویرایش 1.6 و بار دیگر از ورژن 3.0، تسك های مربوط به AsyncTask را به صورت پیش فرض به ترتیب (و پشت سرهم) اجرا می نماید. شما می توانید به سیستم اندروید اعلان نمایید که این تسك ها را به طور موازی با اجرای متد executeOnExecutor()، در حالی که

AsyncTask.THREAD\_POOL\_EXECUTOR به عنوان اولین پارامتر به آن ارسال شده، اجرا نمایید.

تکه کد زیر این عملیات را نمایش می گذارد.

```
// ImageLoader extends AsyncTask
ImageLoader imageLoader = new ImageLoader( imageView );
// Execute in parallel
imageLoader.executeOnExecutor( AsyncTask.THREAD_POOL_EXECUTOR, "http://url.com/image.png" );
```

**نکته:** تغییرات در نحوه پیکربندی و تنظیمات را به صورت خودکار مدیریت نمی کند، بدین معنی که چنانچه activity از نو ساخته شود، برنامه نویس باید آن را با نوشتن کد مربوطه

مدیریت نماید. یک راه حل پرطرفدار و مورد استفاده ای اغلب توسعه دهندگان این است که کلاس fragment) retained headless fragment را در یک AsyncTask را در یک AsynTask که اطلاعات خود را بین تغییرات در نحوه پیکربندی حفظ می کنند) تعریف نمایید.

## مثال: پیاده سازی AsynTask

کد زیر چگونگی استفاده از کلاس AsyncTask جهت دانلود محتواهای یک صفحه وب را به نمایش می گذارد.

یک پروژه ای جدید اندروید به نام activity و یک activity به نام ReadWebpageAsyncTask ایجاد نمایید. مجوز INTERNET را در فایل ReadWebpageAsyncTask تنظیمات اپلیکیشن (manifest) تنظیم نمایید.

زیر را ایجاد نمایید.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <Button
        android:id="@+id/readWebpage"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="onClick"
        android:text="Load Webpage" >
    </Button>
    <TextView
        android:id="@+id/TextView01"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="Placeholder" >
    </TextView>
</LinearLayout>
```

بدنه ای activity خود را به صورت زیر ویرایش نمایید.

```
package de.vogella.android.AsyncTask;
// imports cut out for brevity
public class ReadWebpageAsyncTask extends Activity {
    private TextView textView;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
```

```

        textView = (TextView) findViewById(R.id.TextView01);
    }
    private class DownloadWebPageTask extends AsyncTask<String, Void, String> {
        @Override
        protected String doInBackground(String... urls) {
            // we use the OkHttp library from https://github.com/square/okhttp
            OkHttpClient client = new OkHttpClient();
            Request request =
                new Request.Builder()
                    .url(urls[0])
                    .build();
            Response response = client.newCall(request).execute();
            if (response.isSuccessful()) {
                return response.body().string();
            }
        }
        return "Download failed";
    }
    @Override
    protected void onPostExecute(String result) {
        textView.setText(result);
    }
}
public void onClick(View view) {
    DownloadWebPageTask task = new DownloadWebPageTask();
    task.execute(new String[] { "http://www.vogella.com/index.html" });
}
}

```

اپلیکیشن خود را اجرا نموده و دکمه‌ی مربوطه را کلیک نمایید. اطلاعات صفحه‌ی وب موردنظر در پس زمینه خوانده می‌شود. به مجرد اتمام این فرایند، می‌بینید که آبجکت TextView با اطلاعات جدید بروز آوری می‌شود.

## 24-4-پردازش پس زمینه‌ای و مدیریت چرخه‌ی حیات (lifecycle)

### 24-4-حفظ اطلاعات مربوط به وضعیت (state) بین تغییراتی که در نحوه‌ی

#### پیکربندی رخ می‌دهد

یکی از مشکلات و چالش‌های استفاده از thread، در نظر گرفتن و مدیریت چرخه‌ی حیات اپلیکیشن می‌باشد. سیستم اندروید ممکن است activity را به کلی از بین برده، از حافظه‌ی پاک کند یا با اعمال تغییری در نحوه‌ی پیکربندی سبب شود که activity مجدداً ایجاد گردد.

بعلاوه توسعه دهنده می بایست مدیریت محاوره های (dialog) باز را در نظر بگیرد چرا که این محاوره ها در نهایت به activity میزبان وصل هستند. در صورتی که activity مجدد راه اندازی شده View not شما قصد دسترسی به محاوره ای جاری را بکنید، خواهید دید که یک خطای زمان اجرا attached to window manager از سمت سیستم صادر می شود.

به منظور ذخیره ی یک آبجکت، شما می توانید متدهای onRetainNonConfigurationInstance() را مورد استفاده قرار دهید. در واقع چنانچه قرار است activity مورد نظر از بین رفته و مجدد راه اندازی شود، شما می توانید با فراخوانی متدهای مزبور اطلاعات نمونه ای جاری را در یک آبجکت نگه داشته و دوباره با ساخت activity، داده های ذخیره شده در آبجکت را به آن تحویل دهید. برای بازیابی این آبجکت می توانید متدهای getLastNonConfigurationInstance() را فراخوانی کنید. از این طریق شما قادر خواهید بود، حتی زمانی که activity از بین رفته و مجدد راه اندازی می شود، آبجکت حامل اطلاعات (برای مثال یک thread) در حال اجرا را نگه داشته و محتوای آن را به از نو ساخته شده پاس دهید.

در صورتی که activity برای اولین بار راه اندازی شده و یا با صدا خورده شدن متدهای finish() خاتمه می یابد، متدهای مذکور در خروجی مقدار null را برخواهد گرداند.

البته لازم به توضیح است که متدهای onRetainNonConfigurationInstance() از ورژن 13 کتابخانه های اندروید (API 13) تقریباً منسوخ تلقی می شود. بهتر است برای بدست آوردن نتیجه ی مورد نظر از fragment ها استفاده نموده و متدهای setRetainInstance() را جهت نگه داشتن اطلاعات بین تغییرات در نحوه ی پیکربندی بکار ببرید.

## 24-4- استفاده از آبجکت application جهت ذخیره اطلاعات چندین آبجکت

چنانچه لازم است چندین آبجکت را جهت نگهداری اطلاعات activity ها و بازگردانی آن ها پس از تغییرات در نحوه ی پیکربندی بکار ببرید، در آن صورت می توانید کلاس Application که اطلاعات

مربوط به وضعیت سراسری اپلیکیشن را در خود ذخیره نگه می دارد را در برنامه‌ی خود بیاده سازی نمایید.

برای نیل به این هدف، کافی است مقدار `classname` را به خصیصه در فایل تنظیمات اپلیکیشن خود تخصیص دهید.

```
<application android:icon="@drawable/icon" android:label="@string/app_name"
    android:name="MyApplicationClass">
    <activity android:name=".ThreadsLifecycleActivity"
        android:label="@string/app_name">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
</application>
```

لایه‌ی `runtime` اندروید کلاس `application` را به صورت خودکار ایجاد کرده و تا زمانی که کل فرایند اپلیکیشن متوقف نشده، این آبجکت همچنان در دسترس باقی خواهد ماند.

آبجکت اپلیکیشن می‌تواند اطلاعات مربوط به وضعیت سراسری اپلیکیشن را نگه دارد. این آبجکت قبل از اینکه فرایند مربوط به اپلیکیشن یا پکیج ایجاد شود، نمونه سازی شده و در دسترس قرار می‌گیرد.

از کلاس نام برده می‌توان جهت نگهداری و بازگردانی اطلاعات آبجکت‌هایی که در چندین `activity` هستند یا برای چرخه‌ی حیات کلی اپلیکیشن قابل دسترسی می‌باشد، استفاده نمایید. داخل بدنه‌ی متدهای `onCreate()` می‌توانید آبجکت‌هایی ایجاد کرده و آن‌ها را از طریق فیلدهای `public` یا توابع `getter` در دسترس قرار دهید.

متدهای `onTerminate()` در کلاس اپلیکیشن فقط و فقط به منظور تست بکار می‌رود. همان‌طور که از اسمش پیدا است، این متدهای فرایند تخصیص یافته به اپلیکیشن را به طور کلی از بین می‌برد. بنابراین چنانچه این متدهای فرایند میزبان خاتمه یابد، در آن صورت تمامی منابع اختصاص یافته به اپلیکیشن به صورت خودکار آزاد می‌شوند و اطلاعات از حافظه‌ی دستگاه حذف می‌گردد.

جهت دسترسی به آبجکت Application می توانید متدهای getApplication() را در سطح activity فراخوانی نمایید.

## Fragment-25-4 پردازش موازی در پس زمینه (background processing)

نگهداری اطلاعات آبجکت جهت بازگردانی آن پس از تغییر در نحوه پیکربندی

می توانید از fragment های بدون UI استفاده نموده و اطلاعات آن ها را با فراخوانی متدهای setRetainInstanceState() ذخیره نگه دارید و پس از تغییر در تنظیمات اپلیکیشن آن ها را بازگردانی کنید.

از این طریق آبجکت Thread یا AsyncTask با تغییر در تنظیمات و نحوه پیکربندی اپلیکیشن در حافظه نگه داشته می شود و شما می توانید بدون اینکه نگران مدیریت چرخه ای حیات اپلیکیشن خود بوده و آن را به صورت صریح در نظر بگیرید، عملیاتی را در پس زمینه و در موازات هم به اجرا بگذارید.

## Fragment-25-2 UI های فاقد headless fragments (headless fragments)

اگر پردازش هایی را در پس زمینه به اجرا گذاشته اید، در آن صورت می توانید در زمان اجرا (به صورت dynamic fragment) یک headless fragment به اپلیکیشن خود متصل کرده و setRetainInstanceState() را بر روی true تنظیم نمایید. به دنبال این کار، fragment (اطلاعات آبجکت) با تغییر در نحوه پیکربندی حفظ شده و شما می توانید پردازش ناهمزمان در آن اجرا کنید.

## Loader-26-4

## 4-26-1-هدف از کاربرد کلاس Loader

کلاس Loader به شما این امکان را می دهد تا داده های مورد نیاز را به صورت ناهمزمان (async) در activity یا fragment مربوطه بارگذاری نمایید. این کلاس می تواند منبع داده ها را رصد کرده (زیر نظر داشته) و زمانی که محتويات تغيير کردند، نتایج جديد را تحويل دهد. همچنین اين امكان را فراهم می کند تا داده ها با تغيير در نحوه ی پيکربندی ذخیره شده و در زمان لازم (پس از ساخته شدن fragment) بازگردانی شوند.

داده ها را می توان به وسیله ی Loader در حافظه ی نهان نگه داشت (cache کرد) و زمانی که تغييری در نحوه ی پيکربندی رخ می دهد، داده های ذخیره شده را بازگردانی نمود. کلاس های Loader از اندرويد 3.0 معرفی شده و بخشی از compatibility layer (جهت پشتيبانی از نسخه های قدیمی) برای ورژن 1.6 به بعد اندرويد محسوب می شود.

## 4-26-2-پياده سازی کلاس Loader

مي توانيد از کلاس AsyncTaskLoader به عنوان يك پايه برای پياده سازی های Loader خود استفاده نمایيد. يك LoaderManager می تواند چندین نمونه از Loader را مدیریت کند. نحوه ی فراخوانی Loader در زیر به نمایش گذاشته شده است.

```
# start a new loader or re-connect to existing one
getLoaderManager().initLoader(0, null, this);
```

اولین پارامتر يك ID یا شناسه ی منحصر بفرد است که کلاس callback برای شناسابی Loader مورد استفاده قرار می دهد. دومین پارامتر ارسالی يك آبجکت bundle است که برای اطلاعات بیشتر به متده داده می شود. سومین پارامتر ورودی متده initLoader()، کلاسی است که به محض شروع مقداردهی اولیه (کلاس callback) فراخوانی می شود. اين کلاس می بایست اينترفیس LoaderCallbacks را پياده سازی کند. در واقع مرسوم است activity یا fragment ای که از Loader استفاده می کند، همراه با آن اينترفیس LoaderManager.LoaderCallbacks را نيز پياده سازی نماید.

به طور مستقیم با فراخوانی متد `getLoaderManager().initLoader()` ایجاد نمی شود، بلکه کلاس `callback` یا بازفرایان آن را در بدنه `onCreateLoader()` خود ایجاد می کند.

هنگامی که Loader خواندن داده ها به صورت ناهمزمان را به اتمام می رساند، متد `onLoadFinished()` از کلاس `callback` صدا خورده می شود. در اینجا (داخل بدنه `onCreateLoader()`) شما می تواند UI و ظاهر اپلیکیشن خود را با داده های جدید بروز آوری نمایید.

### 4-24-3- دیتابیس SQLite و پیاده سازی کلاس CursorLoader

چارچوب نرم افزاری اندروید (framework) یک کلاس به نام `CursorLoader` در اختیار توسعه دهنده قرار می دهد که پیاده سازی Loader را به صورت پیش فرض دربرداشته و اتصال به دیتابیس (database connection) را خود مدیریت می نماید.

برای کوئری گرفتن و درخواست داده از یک `ContentProvider` که مبتنی بر دیتابیس SQLite می باشد، برنامه نویسان اغلب از کلاس `CursorLoader` استفاده می کنند. این کلاس Loader در پس زمینه (background thread) از دیتابیس کوئری می گیرد، به همین جهت تعامل اپلیکیشن با کاربر مختل نشده و در نتیجه UI مسدود نمی شود.

کلاس `CursorLoader` جایگزین آجکت های `Activity-managed cursor` که در ویرایش های قبلی اندروید بکار می رفته، محسوب می شود.

در صورتی که Cursor نامعتبر شود، متد `onLoadReset()` در کلاس `callback` فراخوانی می گردد.

**تمرین: پیاده سازی Loader اختصاصی برای مدیریت و بارگذاری ناهمگام**

**جفت های کلید-مقدار (preferences)**

## 4-24-4-پیاده سازی کلاس

در تمرین زیر یک کلاس loader اختصاصی برای بارگذاری و مدیریت جفت های کلید-مقدار preference خواهد کرد. با هر بار بارگذاری (load)، مقدار preferences افزایش می یابد.

یک پروژه ایجاد کنید به نام com.Vogella.android.loader.preferences و همچنین یک activity MainActivity ایجاد نمایید.

با تعریف کلاس زیر که یک پیاده سازی اختصاصی از AsyncTaskLoader هست شما در واقع جفت های کلید-مقدار (shared preferences) را به صورت ناهمگام مدیریت می نمایید (preferences برای ذخیره کوچک بکار می رود تا برای مثال با بسته شدن اپلیکیشن یا خاموش شدن دستگاه اطلاعات مربوطه همچون تنظیمات اپلیکیشن از دست نرود).

```
package com.vogella.android.loader.preferences;
import android.content.AsyncTaskLoader;
import android.content.Context;
import android.content.SharedPreferences;
import android.preference.PreferenceManager;
public class SharedPreferencesLoader extends AsyncTaskLoader<SharedPreferences>
    implements SharedPreferences.OnSharedPreferenceChangeListener {
    private SharedPreferences prefs = null;
    public static void persist(final SharedPreferences.Editor editor) {
        editor.apply();
    }
    public SharedPreferencesLoader(Context context) {
        super(context);
    }
    // Load the data asynchronously
    @Override
    public SharedPreferences loadInBackground() {
        prefs = PreferenceManager.getDefaultSharedPreferences(getContext());
        prefs.registerOnSharedPreferenceChangeListener(this);
        return (prefs);
    }
    @Override
    public void onSharedPreferenceChanged(SharedPreferences sharedPrefs,
                                         String key) {
        // notify loader that content has changed
        onContentChanged();
    }
}
```

```

    /**
     * starts the loading of the data
     * once result is ready the onLoadFinished method is called
     * in the main thread. It loader was started earlier the result
     * is return directly
     * method must be called from main thread.
    */
    @Override
    protected void onStartLoading() {
        if (prefs != null) {
            deliverResult(prefs);
        }
        if (takeContentChanged() || prefs == null) {
            forceLoad();
        }
    }
}

```

نمونه کد زیر استفاده از این loader را در کلاس activity به نمایش می‌گذارد.

```

package com.vogella.android.loader.preferences;
import android.annotation.SuppressLint;
import android.app.Activity;
import android.app.LoaderManager;
import android.content.Loader;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.widget.TextView;
public class MainActivity extends Activity implements
    LoaderManager.LoaderCallbacks<SharedPreferences> {
    private static final String KEY = "prefs";
    private TextView textView;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        textView = (TextView) findViewById(R.id.prefs);
        getLoaderManager().initLoader(0, null, this);
    }
    @Override
    public Loader<SharedPreferences> onCreateLoader(int id, Bundle args) {
        return (new SharedPreferencesLoader(this));
    }
    @SuppressLint("CommitPrefEdits")
    @Override
    public void onLoadFinished(Loader<SharedPreferences> loader,
        SharedPreferences prefs) {
        int value = prefs.getInt(KEY, 0);
        value += 1;
        textView.setText(String.valueOf(value));
        // update value
    }
}

```

```

        SharedPreferences.Editor editor = prefs.edit();
        editor.putInt(KEY, value);
        SharedPreferencesLoader.persist(editor);
    }
    @Override
    public void onLoaderReset(Loader<SharedPreferences> loader) {
        // NOT used
    }
}

```

## تست اپلیکیشن

پس از هر بار تغییر در نحوه ی پیکربندی/تنظیمات اپلیکیشن، کلاس LoaderManager متدهای `onLoadFinished()` و `onLoaderReset()` را به صورت خودکار در `activity` صدا می زند. اپلیکیشن را اجرا نموده و مطمئن شوید که با هر بار تغییر در نحوه ی پیکربندی (config change) مقدار ذخیره شده در `shared preferences` افزایش می یابد.

## 25-4- استفاده از service ها

می توانید از سرویس های اندروید برای اجرای عملیات و ترسک های پس زمینه ای بهره بگیرید. برای این منظور می توانید به مبحث آموزش سرویس ها در اندروید مراجعه نمایید.

### تمرین: چرخه ای حیات thread و activity

نمونه برنامه ی زیر یک فایل تصویری را در `thread` پس زمینه از اینترنت دانلود کرده و یک پنجره ی محاوره به نمایش می گذارد که تا اتمام پروسه ی دانلود در UI باقی می ماند. برنامه را طوری خواهید نوشت که با از بین رفقن و ساخت مجدد `activity` نیز `thread` حفظ شده و پنجره ی محاوره به درستی نمایش داده/بسته شود.

برای این مثال یک پروژه و `activity` جدید به ترتیب به نام های `ThreadsLifecycleActivity` و `de.vogella.android.threadslifecycle` ایجاد نمایید. همچنین لازم است مجوز استفاده از اینترنت را نیز برای اپلیکیشن در فایل `manifest` تنظیم نمایید.

محتویات فایل تنظیمات اپلیکیشن (AndroidManifest.xml) شما می بایست ظاهری مشابه زیر داشته باشد.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="de.vogella.android.threadslifecycle"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk android:minSdkVersion="10" />
    <uses-permission android:name="android.permission.INTERNET" >
    </uses-permission>
    <application
        android:icon="@drawable/icon"
        android:label="@string/app_name" >
        <activity
            android:name=".ThreadsLifecycleActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

محتوای فایل main.xml را به صورت زیر ویرایش نمایید.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <LinearLayout
        android:id="@+id/linearLayout1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >
        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:onClick="downloadPicture"
            android:text="Click to start download" >
        </Button>
        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:onClick="resetPicture"
            android:text="Reset Picture" >
        </Button>
    </LinearLayout>
    <ImageView
        android:id="@+id/imageView1"
        android:layout_width="match_parent"
```

```

    android:layout_height="match_parent"
    android:src="@drawable/icon" >
</ImageView>
</LinearLayout>

```

حال بدنی ی کلاس activity را ویرایش نمایید. داخل این کلاس thread ذخیره شده و محاوره به محض از بین رفتن activity، بسته می شود.

```

package de.vogella.android.threadslifecycle;
import java.io.IOException;
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.StatusLine;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpUriRequest;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.util.EntityUtils;
import android.app.Activity;
import android.app.ProgressDialog;
import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.view.View;
import android.widget.ImageView;
public class ThreadsLifecycleActivity extends Activity {
    // Static so that the thread access the latest attribute
    private static ProgressDialog dialog;
    private static Bitmap downloadBitmap;
    private static Handler handler;
    private ImageView imageView;
    private Thread downloadThread;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        // create a handler to update the UI
        handler = new Handler() {
            @Override
            public void handleMessage(Message msg) {
                imageView.setImageBitmap(downloadBitmap);
                dialog.dismiss();
            }
        };
        // get the latest imageView after restart of the application
    }
}

```

```

imageView = (ImageView) findViewById(R.id.imageView1);
Context context = imageView.getContext();
System.out.println(context);
// Did we already download the image?
if (downloadBitmap != null) {
    imageView.setImageBitmap(downloadBitmap);
}
// check if the thread is already running
downloadThread = (Thread) getLastNonConfigurationInstance();
if (downloadThread != null && downloadThread.isAlive()) {
    dialog = ProgressDialog.show(this, "Download", "downloading");
}
}

public void resetPicture(View view) {
    if (downloadBitmap != null) {
        downloadBitmap = null;
    }
    imageView.setImageResource(R.drawable.icon);
}

public void downloadPicture(View view) {
    dialog = ProgressDialog.show(this, "Download", "downloading");
    downloadThread = new MyThread();
    downloadThread.start();
}

// save the thread
@Override
public Object onRetainNonConfigurationInstance() {
    return downloadThread;
}

// dismiss dialog if activity is destroyed
@Override
protected void onDestroy() {
    if (dialog != null && dialog.isShowing()) {
        dialog.dismiss();
        dialog = null;
    }
    super.onDestroy();
}

// Utiliy method to download image from the internet
static private Bitmap downloadBitmap(String url) throws IOException {
    HttpUriRequest request = new HttpGet(url);
    HttpClient httpClient = new DefaultHttpClient();
    HttpResponse response = httpClient.execute(request);
    StatusLine statusLine = response.getStatusLine();
    int statusCode = statusLine.getStatusCode();
    if (statusCode == 200) {
        HttpEntity entity = response.getEntity();
        byte[] bytes = EntityUtils.toByteArray(entity);
        Bitmap bitmap = BitmapFactory.decodeByteArray(bytes, 0,
            bytes.length);
        return bitmap;
    } else {
}
}

```

```

        throw new IOException("Download failed, HTTP response code "
            + statusCode + " - " + statusLine.getReasonPhrase());
    }
}

static public class MyThread extends Thread {
    @Override
    public void run() {
        try {
            // Simulate a slow network
            try {
                new Thread().sleep(5000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            downloadBitmap =
downloadBitmap("http://www.devoxx.com/download/attachments/4751369/DV11");
            // Updates the user interface
            handler.sendMessage(0);
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
        }
    }
}

```

اپلیکیشن خود را اجرا نموده و جهت راه اندازی فرایند دانلود، بر روی دکمه `i` start کلیک نمایید. برای اینکه مطمئن شوید رفتار اپلیکیشن در خصوص مدیریت lifecycle یا چرخه `i` حیات صحیح می باشد، می توانید در محیط شبیه ساز با فشردن کلید های `ctrl+F11` جهت و وضعیت (orientation) نمایش را تغییر دهید.

لازم به ذکر است که `Thread` یک کلاس `static` و `inner` (ایستا و تعریف شده در دل یک کلاس دیگر) می باشد. به طور کلی لازم است برای پردازش های پس زمینه ای یک کلاس `static inner` را بکار ببرید چرا که در غیر این صورت کلاس `inner` (درون ساخته در دل کلاس دیگر) اشاره گری به کلاسی که در آن تعریف شده را در خود نگه می دارد. زمانی که `thread` به نمونه `activity` از ارسال می شود، از آنجایی که این `thread` همچنان به نمونه `i` قبلی اشاره دارد، هدر رفت حافظه و `memory leak` رخ خواهد داد.



بخش هشتم :

## JSON-26-4 و اندروید

در این آموزش نحوه‌ی پردازش JSON در اندروید را خواهید آموخت.

449

آدرس آموزشگاه : تهران - خیابان شریعتی - بالاتر از خیابان ملک - جنب بانک صادرات - پلاک 651 طبقه دوم - واحد 7

88146323 - 88446780 - 88146330

## 4-26-1-کتابخانه ی پیش فرض و درون ساخته ی اندروید برای پردازش

### JSON

محیط (platform) اندروید خود یک کتابخانه ی درون ساخته به نام json.org ویژه ی پردازش و ساخت فایل های JSON دارد. در صورت تمایل شما می توانید سایر کتابخانه های کد باز نظیر Moshi یا Gson را برای پردازش و ساخت JSON مورد استفاده قرار دهید.

### مثال: خواندن فرمت JSON

تبدیل یک رشته ی JSON به آبجکت JSON بسیار ساده است.

کافی است کد زیر را برای activity خود درج نمایید.

```
import org.json.JSONArray;
import org.json.JSONObject;
String jsonString = readJsonObjectFromSomeWhere();
try {
    JSONObject json = new JSONObject(jsonString);
} catch (Exception e) {
    e.printStackTrace();
}
```

1

متدی است که یک رشته از جنس JSON را می خواند. به منظور کوتاهی کد معمولاً از نوشتن آن خودداری می شود.

**نکته:** نمونه ی کد فوق در thread اصلی قابل اجرا نیست. لازم است این تکه کد را خارج از thread اصلی اجرا نمایید.

## 4-26-2-ساخت JSON

نوشتن JSON بسیار ساده است. کافی است یک JSONArray یا JSONObject ایجاد نموده و متد `toString()` را فراخوانی نمایید.

```
public void writeJSON() {
    JSONObject object = new JSONObject();
    try {
        object.put("name", "Jack Hack");
```

450

```
        object.put("score", new Integer(200));
        object.put("current", new Double(152.32));
        object.put("nickname", "Hacker");
    } catch (JSONException e) {
        e.printStackTrace();
    }
    System.out.println(object);
}
```





452

آدرس آموزشگاه : تهران - خیابان شریعتی - بالاتر از خیابان ملک - جنب بانک صادرات - پلاک 651 طبقه دوم - واحد 7

88146323 - 88446780 - 88146330

## بخش اول :

# پیاده سازی قابلیت drag&drop (کشیدن و جایگذاری)/کار با drag and drop

### 1-5-استفاده از drag & drop در اندروید

قابلیت drag & drop (کشیدن و جایگذاری) یک view بر روی view group یا view های دیگر از ویرایش 4.0 اندروید پشتیبانی می شود.

#### 1-5-1-پیاده سازی قابلیت کشیدن view

به منظور پیاده سازی قابلیت drag، شما می بایست OnTouchListener یا LongClickListener interface را بر روی view ای که قرار است کشیده یا drag شود، تنظیم نمایید. به عبارت دیگر یکی از این دو را پیاده سازی کرده و متدهای آن را در کلاس بازنویسی کنید.

متدهای startDrag و کشیده شدن آیتم را آغاز می کند. در این متدهای startDrag view یا مکانی که view جاری قرار است بر روی آن کشیده و جایگذاری شود را به واسطه ای نمونه ای از کلاس ClipData مشخص نمایید.

سپس می بایست نمونه ای از کلاس DragShadowBuilder را به عنوان آرگومان به متدهای startDrag و endDrag ارسال نمایید. این آبجکت عکسی که برای عملیات drag استفاده می شود را مشخص می نماید (به عبارت دیگر با استفاده از این آبجکت می توان برای آیتم در حال drag سایه ایجاد کنید). حال می

توانید این view را مستقیماً به عنوان پارامتر ارسال کنید که در این صورت تصویری از view مورد نظر در طول کشیده شدن آیتم نمایش داده می‌شود.

نحوه‌ی تنظیم و راه اندازی عملیات کشیدن در touch listener با کد زیر به صورت عملی به نمایش گذاشته شده است.

```
// Assign the touch listener to your view which you want to move
findViewById(R.id.myimage1).setOnTouchListener(new MyTouchListener());
// This defines your touch listener
private final class MyTouchListener implements OnTouchListener {
    public boolean onTouch(View view, MotionEvent motionEvent) {
        if (motionEvent.getAction() == MotionEvent.ACTION_DOWN) {
            ClipData data = ClipData.newPlainText("", "");
            DragShadowBuilder shadowBuilder = new View.DragShadowBuilder(
                view);
            view.startDrag(data, shadowBuilder, view, 0);
            view.setVisibility(View.INVISIBLE);
            return true;
        } else {
            return false;
        }
    }
}
```

## 5-1-2- تعریف مشخصات و اطلاعات محل جایگذاری view (مشخص کردن drop target)

هایی که قرار است از قابلیت drop پشتیبانی کند، می‌بایست نمونه‌ای از کلاس OnDragListener به آن‌ها اختصاص یابد. به عبارت دیگر، لازم است نمونه‌ای از onDragListener را به view ای که قرار است محل جایگذاری view دیگر باشد، تخصیص دهید.

در این گوش فراخوان به رویداد drop (listener)، چنانچه از قبل event‌های مرتبط با drag&drop تعریف شده باشد، callback یا توابع بازفراروانی را دریافت خواهید کرد.

DragEvent.ACTION_DRAG_STARTED	*	DragEvent.ACTION_DRAG_ENTERED	*
DragEvent.ACTION_DRAG_EXITED	*	DragEvent.ACTION_DROP	*
		. DragEvent.ACTION_DRAG_ENDED	

ای که View برای آن پیاده سازی شده در واقع محل قرارگیری یک view دیگر بوده و از قابلیت drop پشتیبانی می کند. این setOnDragListener است که نمونه ای از آن را به این آبجکت تخصیص می دهد.

```
findViewById(R.id.bottomright).setOnDragListener(new MyDragListener());
class MyDragListener implements OnDragListener {
    Drawable enterShape = getResources().getDrawable(
        R.drawable.shape_droptarget);
    Drawable normalShape = getResources().getDrawable(R.drawable.shape);
    @Override
    public boolean onDrag(View v, DragEvent event) {
        int action = event.getAction();
        switch (event.getAction()) {
            case DragEvent.ACTION_DRAG_STARTED:
                // do nothing
                break;
            case DragEvent.ACTION_DRAG_ENTERED:
                v.setBackgroundDrawable(enterShape);
                break;
            case DragEvent.ACTION_DRAG_EXITED:
                v.setBackgroundDrawable(normalShape);
                break;
            case DragEvent.ACTION_DROP:
                // Dropped, reassign View to ViewGroup
                View view = (View) event.getLocalState();
                ViewGroup owner = (ViewGroup) view.getParent();
                owner.removeView(view);
                LinearLayout container = (LinearLayout) v;
                container.addView(view);
                view.setVisibility(View.VISIBLE);
                break;
            case DragEvent.ACTION_DRAG_ENDED:
                v.setBackgroundDrawable(normalShape);
            default:
                break;
        }
        return true;
    }
}
```

## تمرین: پیاده سازی Drag and drop در اپلیکیشن به صورت عملی

در این تمرین شما تعدادی view group تعریف می کنید که از قابلیت drag&drop (کشیدن آیتم از یکی و جایگذاری آن در دیگری) پشتیبانی می کند.

## ایجاد پروژه

یک پروژه و com.vogella.android.draganddrop activity به نام های activity و com.vogella.android.draganddrop به ترتیب به نام های DragActivity و DragActivity ایجاد نمایید.

## ۱-۵-۱-۳- ایجاد فایل های Drawable (فایل های تصویری تعریف شده در فرمت XML)

در این بخش یکی از منابع مورد استفاده ی شما، فایل های XML drawable خواهند بود.

بنابراین ابتدا بایستی فایل های XML drawable مورد نیاز پروژه ی خود را در پوشه res/drawable ایجاد نمایید.

اولین فایلی که در آدرس مذکور ایجاد می کنید، shape.xml با محتویات زیر خواهد بود.

```
<?xml version="1.0" encoding="UTF-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle" >
    <stroke
        android:width="2dp"
        android:color="#FFFFFF" />
    <gradient
        android:angle="225"
        android:endColor="#DD2ECCFA"
        android:startColor="#DD000000" />
    <corners
        android:bottomLeftRadius="7dp"
        android:bottomRightRadius="7dp"
        android:topLeftRadius="7dp"
        android:topRightRadius="7dp" />
</shape>
```



سپس فایل shape\_droptarget.xml را با محتویات زیر در پوشه res ایجاد نمایید.

```
<?xml version="1.0" encoding="UTF-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle" >
    <stroke
        android:width="2dp"
        android:color="#FFFF0000" />
    <gradient
        android:angle="225"
        android:endColor="#DD2ECCFA"
        android:startColor="#DD000000" />
    <corners
        android:bottomLeftRadius="7dp"
        android:bottomRightRadius="7dp"
        android:topLeftRadius="7dp"
        android:topRightRadius="7dp" />
</shape>
```

&lt;/shape&gt;

## Activity-5-1-4 و تنظیم فایل layout

محفویات activity کلاس layout خود را به صورت زیر ویرایش نمایید.

```
<?xml version="1.0" encoding="utf-8"?>
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:columnCount="2"
    android:columnWidth="320dp"
    android:orientation="vertical"
    android:rowCount="2"
    android:stretchMode="columnWidth" >
    <LinearLayout
        android:id="@+id/topleft"
        android:layout_width="160dp"
        android:layout_height="160dp"
        android:layout_row="0"
        android:background="@drawable/shape" >
        <ImageView
            android:id="@+id/myimage1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@drawable/ic_launcher" />
    </LinearLayout>
    <LinearLayout
        android:id="@+id/topright"
        android:layout_width="160dp"
        android:layout_height="160dp"
        android:background="@drawable/shape" >
        <ImageView
            android:id="@+id/myimage2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@drawable/ic_launcher" />
    </LinearLayout>
    <LinearLayout
        android:id="@+id/bottomleft"
        android:layout_width="160dp"
        android:layout_height="160dp"
        android:background="@drawable/shape" >
        <ImageView
            android:id="@+id/myimage3"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@drawable/ic_launcher" />
    </LinearLayout>
    <LinearLayout
        android:id="@+id/bottomright"
```

```

    android:layout_width="160dp"
    android:layout_height="160dp"
    android:background="@drawable/shape" >
    <ImageView
        android:id="@+id/myimage4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/ic_launcher" />
</LinearLayout>
</GridLayout>
```

حال بدنی کلاس activity خود را به صورت زیر ویرایش نمایید.

```

package com.vogella.android.draganddrop;
import android.app.Activity;
import android.content.ClipData;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import android.view.DragEvent;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.DragShadowBuilder;
import android.view.View.OnDragListener;
import android.view.View.OnTouchListener;
import android.view.ViewGroup;
import android.widget.LinearLayout;
public class DragActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        findViewById(R.id.myimage1).setOnTouchListener(new MyTouchListener());
        findViewById(R.id.myimage2).setOnTouchListener(new MyTouchListener());
        findViewById(R.id.myimage3).setOnTouchListener(new MyTouchListener());
        findViewById(R.id.myimage4).setOnTouchListener(new MyTouchListener());
        findViewById(R.id.topleft).setOnDragListener(new MyDragListener());
        findViewById(R.id.topright).setOnDragListener(new MyDragListener());
        findViewById(R.id.bottomleft).setOnDragListener(new MyDragListener());
        findViewById(R.id.bottomright).setOnDragListener(new MyDragListener());
    }
    private final class MyTouchListener implements OnTouchListener {
        public boolean onTouch(View view, MotionEvent motionEvent) {
            if (motionEvent.getAction() == MotionEvent.ACTION_DOWN) {
                ClipData data = ClipData.newPlainText("", "");
                DragShadowBuilder shadowBuilder = new View.DragShadowBuilder(
                        view);
                view.startDrag(data, shadowBuilder, view, 0);
                view.setVisibility(View.INVISIBLE);
                return true;
            } else {

```

```

        return false;
    }
}
}

class MyDragListener implements OnDragListener {
    Drawable enterShape = getResources().getDrawable(
        R.drawable.shape_droptarget);
    Drawable normalShape = getResources().getDrawable(R.drawable.shape);

    @Override
    public boolean onDrag(View v, DragEvent event) {
        int action = event.getAction();
        switch (event.getAction()) {
            case DragEvent.ACTION_DRAG_STARTED:
                // do nothing
                break;
            case DragEvent.ACTION_DRAG_ENTERED:
                v.setBackgroundDrawable(enterShape);
                break;
            case DragEvent.ACTION_DRAG_EXITED:
                v.setBackgroundDrawable(normalShape);
                break;
            case DragEvent.ACTION_DROP:
                // Dropped, reassign View to ViewGroup
                View view = (View) event.getLocalState();
                ViewGroup owner = (ViewGroup) view.getParent();
                owner.removeView(view);
                LinearLayout container = (LinearLayout) v;
                container.addView(view);
                view.setVisibility(View.VISIBLE);
                break;
            case DragEvent.ACTION_DRAG_ENDED:
                v.setBackgroundDrawable(normalShape);
            default:
                break;
        }
        return true;
    }
}
}

```

پس از راه اندازی این activity، می توانید `ImageViews` را کشیده و در ظرف دیگری جایگذاری نمایید.



460

آدرس آموزشگاه : تهران - خیابان شریعتی - بالاتر از خیابان ملک - جنب بانک صادرات - پلاک 651 طبقه دوم - واحد 7

88146323 - 88446780 - 88146330

## بخش دوم :

# کار با **drawable** ها / فایل های تصویری تعریف شده در XML

این آموزش به شرح مفهوم Drawable ها و نحوه استفاده از آن ها در اندروید می پردازد.

## Drawable-2-5 چیست؟

یک مفهوم کلی در اندروید برای هر فایل گرافیکی قابل ترسیم می باشد یا به عبارت دیگر drawable هر چیزی است که بتوان آن را کشید. به عنوان ساده ترین نمونه می توان به یک فایل گرافیکی (bitmap) اشاره کرد که در اندروید توسط کلاس BitmapDrawable در اختیار توسعه دهنده قرار می گیرد.

هر drawable در قالب یک فایل مجزا در مسیر res/drawable ذخیره می شود. معمولاً فایل های drawable در قالب فایل های bitmap در وضوح مختلف داخل زیرپوشه های -hdpi ، -mdpi xxhdpi – xhdpi مسیر res/drawable ذخیره می شوند. ویژارد و راهنمای ساخت پروژه ای ADT این پوشه ها را خود به صورت پیش فرض ایجاد می کند. اگر bitmap ها در پوشه ای متفاوتی ذخیره شده باشند، سیستم اندروید خود تنظیمات دستگاه و اندازه ای صفحه نمایش را ارزیابی کرده و بر اساس آن گزینه ای مناسب را انتخاب می کند.

چنانچه عکس مورد نظر را در تمامی وضوح (تراکم پیکسلی و با کیفیت مناسب) نداشته باشید، در آن صورت سیستم اندروید تصویر مربوطه را جهت سازگاری کامل با صفحه نمایش دستگاه میزبان کوچک یا بزرگ می کند. البته این قابلیت ظاهر چندان مناسبی را به وجود نمی آورد چرا که در بیشتر موارد عکس مورد نظر تار شده و از وضوح و کیفیت مناسب برخوردار نخواهد بود.

علاوه بر فایل های گرافیکی، اندروید از drawable های مبتنی بر XML و 9-patch پشتیبانی می کند. drawable های که در فایل های XML تعریف می شوند به برنامه نویس این امکان را می دهد تا رنگ، حاشیه، شبکه/طیف رنگ و گوشش ها را در قالب تگ shape و همچنین state (جهت تعریف عکس خاص برای هر وضعیت ای که view در آن قرار می گیرد)، transition (برای تعریف انیمیشن) فایل گرافیکی مورد نظر را تعریف کند.

به وسیله ی فایل های گرافیکی 9-patch می توان مشخص کرد که در صورت بزرگ تر بودن view میزبان، کدام بخش از فایل گرافیکی مورد نظر می باشد کشیده (بزرگتر) شود.

فایل های Drawable را می توان با کدهای Java نوشت. هر آبجکتی که توابع اینترفیس Drawable را پیاده سازی می کند را می توان به صورت فایل ترسیم شونده/Drawable در کد بکار برد.

### 3-5-استفاده از view ها در drawable ها

جهت دسترسی و اشاره به drawable ها در XML می توان از این ساختار استفاده نمود: @drawable/filename. در این ساختار نگارشی filename همان اسم فایل بدون پسوند می باشد. برای مثال جهت دستیابی به فایل drawable در مسیر res/drawable/hello.png و درج آن در پس زمینه UI، کافی است ساختار @drawable/hello را به صورت زیر استفاده نمایید.

```
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/textView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="@drawable/hello"
    android:text="@string/hello_world" />
```

می توان با کدنویسی drawable ها را به view ها اختصاص داد. بیشتر view ها این قابلیت را دارند که ID فایل منبع مورد نظر را به عنوان پارامتر ورودی بپذیرند. کد زیر نحوه ای تخصیص یک فایل drawable را به عنوان تصویر پس زمینه به ImageView نمایش می دهد.

```
ImageView imageView = (ImageView) findViewById(R.id.image);
imageView.setImageResource(R.drawable.hello);
```

## 4-5-بارگذاری Bitmap ها و drawable ها

جهت استفاده از bitmap ها در اندروید می توانید از کلاس Bitmap استفاده نمایید. در این بخش خواهید آموخت چگونه با کد جاوا آبجکت های Bitmap ایجاد نموده و سپس آن را به آبجکت های Drawable و بالعکس تبدیل نمایید.

این امکان برای شما وجود دارد که فایل bitmap موجود را با کدنویسی بارگذاری نموده و متعاقباً آن ها را به آبجکت های Drawable تبدیل کرد.

مثال زیر نمایش می دهد چگونه می توان یک آبجکت Bitmap در پوشه‌ی assets ایجاد نموده و آن را به المان رابط کاربری ImageView تخصیص داد.

```
AssetManager manager = getAssets();
// read a Bitmap from Assets
InputStream open = null;
try {
    open = manager.open("logo.png");
    Bitmap bitmap = BitmapFactory.decodeStream(open);
    // Assign the bitmap to an ImageView in this layout
    ImageView view = (ImageView) findViewById(R.id.imageView1);
    view.setImageBitmap(bitmap);
} catch (IOException e) {
    e.printStackTrace();
} finally {
    if (open != null) {
        try {
            open.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

همچنین می توانید به فایل های Drawable از پوشه‌ی res/drawable به صورت آبجکت های Bitmap در کد برنامه دسترسی پیدا کنید. کد زیر این قابلیت را به نمایش می گذارد.

```
Bitmap b = BitmapFactory.decodeResource(getResources(), R.drawable.ic_action_search);
```

تعریف نمایید.

```
Bitmap originalBitmap = <initial setup>;
Bitmap resizedBitmap =
    Bitmap.createScaledBitmap(originalBitmap, newWidth, newHeight, false);
```

به منظور تبدیل یک آبجکت Bitmap می توانید کد زیر را بکار ببرید.

```
# Convert Bitmap to Drawable
Drawable d = new BitmapDrawable(getResources(),bitmap);
```

## XML های مبتنی بر Drawable-5-5

### Shape-5-5-1

ها نیز یک نوع drawable هستند که در فرمت XML تعریف شده و به توسعه دهنده این امکان را می دهند تا آبجکت یا اشکال هندسی با رنگ، حاشیه و شبیه رنگ تعریف نمایند.

می توانید این اشکال هندسی را به view های مورد نظر تخصیص داده و از آن ها در المان های رابط کاربری استفاده نمایید.

مزیت استفاده از shape در این است که خود را به صورت اتوماتیک به مقیاس مناسب تنظیم می کند.

کد زیر نمونه ای از یک Shape را نشان می دهد.

```
<?xml version="1.0" encoding="UTF-8"?>
<shape
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
    <stroke
        android:width="2dp"
        android:color="#FFFFFF" />
    <gradient
        android:endColor="#DDBBBBBB"
        android:startColor="#DD777777"
        android:angle="90" />
    <corners
        android:bottomRightRadius="7dp"
        android:bottomLeftRadius="7dp"
        android:topLeftRadius="7dp"
        android:topRightRadius="7dp" />
```

&lt;/shape&gt;

می توانید drawable مزبور را به ویژگی (property) background در فایل layout تخصیص داده و آن را به عنوان تصویر پس زمینه ی UI تنظیم نمایید.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/myshape"
    android:orientation="vertical" >
    <EditText
        android:id="@+id/editText1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        >
    </EditText>
    <RadioGroup
        android:id="@+id/radioGroup1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >
        <RadioButton
            android:id="@+id/radio0"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:checked="true"
            android:text="@string/celsius" >
        </RadioButton>
        <RadioButton
            android:id="@+id/radio1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/fahrenheit" >
        </RadioButton>
    </RadioGroup>
    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/calc"
        android:onClick="myClickHandler">
    </Button>
</LinearLayout>
```

## state های مبتنی بر Drawable-5-5-2

می توان برای هر وضعیت یک drawable ویژه تعریف کرد و سپس بسته به وضعیت جاری، drawable مربوطه را به view تخصیص داد. برای مثال تکه کد زیر وضعیت دکمه را در نظر می گیرد و سپس با توجه به وضعیت آن، drawable مربوطه را به view اختصاص می دهد.

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@drawable/button_pressed"
          android:state_pressed="true" />
    <item android:drawable="@drawable/button_checked"
          android:state_checked="true" />
    <item android:drawable="@drawable/button_default" />
</selector>
```

## Drawble های که طی انتقال جایگزین دیگری می شوند (transition drawble)-5-5-3

شما می توانید در تگ transition، افکت انتقال و جابجایی را تعریف کرده و آن را در کد فعال نمایید. از این طریق یک تصویر طی transition جایگزین تصویری قبلی می شود.

```
<?xml version="1.0" encoding="utf-8"?>
<transition xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@drawable/first_image" />
    <item android:drawable="@drawable/second_image" />
</transition>

final ImageView image = (ImageView) findViewById(R.id.image);
final ToggleButton button = (ToggleButton) findViewById(R.id.button);
button.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(final View v) {
        TransitionDrawable drawable = (TransitionDrawable) image.getDrawable();
        if (button.isChecked()) {
            drawable.startTransition(500);
        } else {
            drawable.reverseTransition(500);
        }
    }
});
```

## 6-5 Drawable های برداری/توسعه پذیر بدون از دست رفت (vector drawable) کیفیت

از ویرایش 5.0، اندروید به توسعه دهنگان این امکان را می دهد تا Drawable های برداری/vector مانند فایل های SVG تعریف کند. کد زیر کاربرد آن را در فایل vectordrawable.xml به نمایش می گذارد. استفاده از Drawable های برداری این مزیت را دارد که خود را با توجه به تراکم پیکسلی و کیفیت دستگاه میزان تنظیم می کند.



```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:height="64dp"
    android:width="64dp"
    android:viewportHeight="600"
    android:viewportWidth="600" >
    <group
        android:name="rotationGroup"
        android:pivotX="300.0"
        android:pivotY="300.0"
        android:rotation="45.0" >
        <path
            android:name="v"
            android:fillColor="#000000"
            android:pathData="M300,70 L 0,-70 70,70 0,0 -70,70z" />
    </group>
</vector>
```

از ویرایش 5.0، کلاس جدیدی به نام `AnimatedVectorDrawable` به کتابخانه های اندروید اضافه شد که به برنامه نویس اجازه می دهد تا Drawable های برداری را با انیمیشن ترکیب کند (ویژگی های Drawable را با انیمیشن های تعریف شده توسط `AnimatorSet` یا `ObjectAnimator` متحرک و پویا کند). برای مشاهده ی مثال هایی بیشتر می توانید به آدرس <http://blog.sqisland.com/2014/10/first-look-at-animated-vector-drawable.html> مراجعه نمایید.

برای استفاده از Drawable های برداری در platform یا ورژن های قدیمی تر چارچوب نرم افزاری اندروید، می توانید از کتابخانه `Compat` (`VectorDrawableCompat`) کتابخانه که امکان استفاده از Drawable های برداری در ورژن های قبلی اندروید را فراهم می آورد) استفاده نمایید.

## 7-5 Drawable animation (تعریف انیمیشن با بارگذاری یک پس از drawable)

می توانید animation drawable تعریف کرده و با استفاده از متده استفاده از متد setBackgroundResource() می توانید View تخصیص دهید. اندروید به شما این امکان را می دهد تا drawable هایی را با حالت انیمیشن یکی پس از دیگری نمایش دهید.

```
<!-- Animation frames are phase*.png files inside the
res/drawable/ folder -->
<animation-list android:id="@+id/selected" android:oneshot="false">
    <item android:drawable="@drawable/phase1" android:duration="400" />
    <item android:drawable="@drawable/phase2" android:duration="400" />
    <item android:drawable="@drawable/phase3" android:duration="400" />
</animation-list>
ImageView img = (ImageView) findViewById(R.id.yourid);
img.setBackgroundDrawable(R.drawable.your_animation_file);
// Get the AnimationDrawable object.
AnimationDrawable frameAnimation = (AnimationDrawable) img.getBackground();
// Start the animation (looped playback by default).
frameAnimation.start();
```

همچنین می توانید آبجکتی پیاده سازی کنید که از کلاس Drawable ارث بری کرده و توابع اینترفیس Animatable را پیاده سازی می کند.

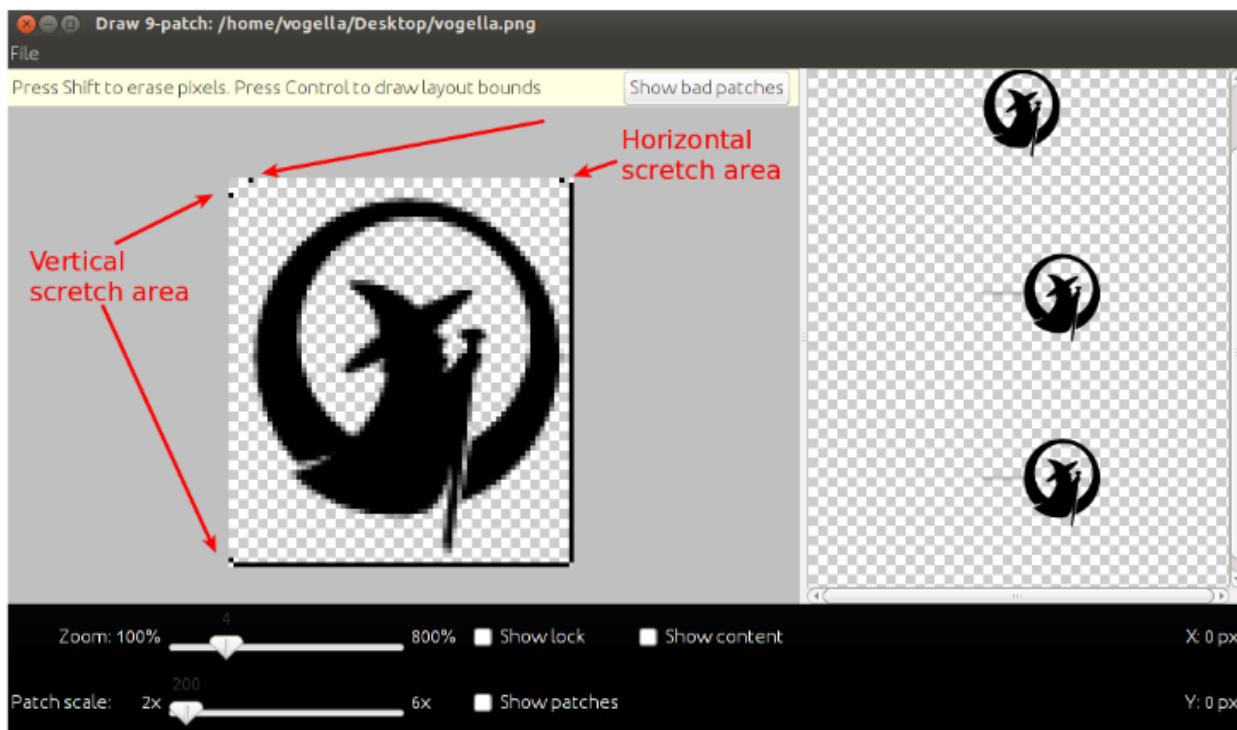
های برداری را می توان با استفاده از کتابخانه VectorDrawableCompat در ورژن Drawable های قبلي های اندروید نيز بكار برد.

## 5-7-1 nine-patch (فایل های ترسیم شونده ی منعطف با کناره های بسط پذیر)

تصور کنید یک عکس دارید و زمانی که شما این تصویر را می کشید، کناره یا لبه ی عکس تار می شود. حال اگر شما تصویری داشته باشید که وسط آن تغییر نکرده و ثابت بماند (با کشیدن تصویر تار نشود)، می توانید آن بخش هایی که با بزرگتر شدن عکس مات نمی شوند را 9-patch تعریف کنید. سیستم اندروید در زمان اجرای برنامه بخش های علامت گذاری نشده را از نظر اندازه تغییر

نمی دهد، اما سایر بخش ها را کش داده و عکس را بزرگ می کند. بدین وسیله زمانی که تصویر برای کاربر به نمایش در می آید، کیفیت آن کاهش نمی یابد.

فایل های 9-patch Drawable حاشیه دارند. در بالا و سمت چپ می توانید ناحیه ای تعریف کنید که متناسب با view اندازه بندی و در صورت لزوم بزرگ می شوند. این ناحیه stretch area است.



چنانچه drawable قرار است در یک view اجازه نوشتن بر روی آن را داشته باشد (همچون آبجکت Button)، در آن صورت می توانید در کناره های سمت راست و پایین ناحیه ای تعریف کنید که در آن متن امکان درج را داشته باشد.

برنامه ای به نام android-sdk/tools ارائه می دهد که ساخت Drawable های انعطاف پذیر و 9-patch را آسان می سازد.

## Drawable های اختصاصی 8-5

می توانید drawable های اختصاصی تعریف کنید که برای نمایش در UI از Canvas API بهره می گیرند. می توانید با استفاده از تمامی توابع/Canvas API فایل های ترسیم شونده یا drawable را متناسب با نیاز خود تنظیم و طراحی کنید.

## 9-5-ساخت های اختصاصی drawable

پروژه جدید به نام com.vogella.android.drawables.custom ایجاد کرده و قالب آماده ی Empty Activity را برای توسعه ی پروژه ی خود انتخاب نمایید. کلاس اختصاصی Drawable را به صورت زیر پیاده سازی نمایید.

```
package com.vogella.android.drawables.custom;
import android.graphics.Bitmap;
import android.graphics.BitmapShader;
import android.graphics.Canvas;
import android.graphics.ColorFilter;
import android.graphics.Paint;
import android.graphics.PixelFormat;
import android.graphics.RectF;
import android.graphics.Shader;
import android.graphics.drawable.Drawable;
public class MyRoundCornerDrawable extends Drawable {
    private Paint paint;
    public MyRoundCornerDrawable(Bitmap bitmap) {
        BitmapShader shader;
        shader = new BitmapShader(bitmap, Shader.TileMode.CLAMP,
                Shader.TileMode.CLAMP);
        paint = new Paint();
        paint.setAntiAlias(true);
        paint.setShader(shader);
    }
    @Override
    public void draw(Canvas canvas) {
        int height = getBounds().height();
        int width = getBounds().width();
        RectF rect = new RectF(0.0f, 0.0f, width, height);
        canvas.drawRoundRect(rect, 30, 30, paint);
    }
    @Override
    public void setAlpha(int alpha) {
        paint.setAlpha(alpha);
    }
    @Override
    public void setColorFilter(ColorFilter cf) {
        paint.setColorFilter(cf);
    }
}
```

```

    }
    @Override
    public int getOpacity() {
        return PixelFormat.TRANSLUCENT;
    }
}
}

```

برای استفاده از این کلاس در متن پروژه، فایل layout خود را به صورت زیر ویرایش نمایید.

```

<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" >
    <ImageView
        android:id="@+id/image"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:contentDescription="TODO" />
</RelativeLayout>

```

بیاده سازی کلاس MainActivity خود را به صورت زیر ویرایش نمایید. کد فرض را بر این می گذارد شما یک فایل bitmap به نام dog.png در پوشه‌ی drawable خود دارید.

```

package com.vogella.android.drawables.custom;
import java.io.InputStream;
import android.app.Activity;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.Bundle;
import android.view.Menu;
import android.widget.ImageView;
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ImageView button = (ImageView) findViewById(R.id.image);
        InputStream resource = getResources().openRawResource(R.drawable.dog);
        Bitmap bitmap = BitmapFactory.decodeStream(resource);
        button.setBackground(new MyRoundCornerDrawable(bitmap));
    }
}

```



بخش سوم :

## طراحی اپلیکیشن بر اساس اصول و قواعد مشخص/با استفاده از theme ها و style ها

این آموزش اصول طراحی بهینه‌ی اپلیکیشن را برای شما تشریح می‌کند. سپس شرح می‌دهد چگونه می‌توانید در اپلیکیشن‌های خود style و theme ایجاد کرده و بکار ببرید.

### 5-10- اصول طراحی UI در اندروید

جهت تعریف UI کارامد برای اپلیکیشن‌های خود بهتر است از اصول زیر پیروی نمایید. این اصول به تفصیل تحت آدرس <http://developer.android.com/design/index.html> شرح داده شده اند.

1. UI را طوری طراحی کنید که با توجه به قابلیت لمس بهینه و کارآمد باشد.
  2. تنها آنچه لازم است نمایش دهید.
  3. لزومی ندارد برای انجام هر عملیاتی از کاربر اجازه بگیرید، با این وجود لازم است امکان لغو عملیات یا بازگرداندن آن را برای کاربر مهیا کنید.
  4. تنها در صورت لزوم تعامل کاربر با UI را مختل نمایید.
  5. پیغام ها را تا حد امکان مختصر نگه داشته و از عکس برای رساندن مفهوم مورد نظر استفاده نمایید.
  6. برنامه و ظاهر آن را طوری طراحی نمایید که اطلاعات کاربر به طور امن در آن نگهداری شده و هیچگاه از دست نرود.
  7. به کاربران این امکان را بدهید تا آیتم های لازم را سریع ایجاد کنند.
  8. اگر ظاهر یکسان است، در آن صورت رفتار نیز باید یکسان باشد.
  9. در تصمیم گیری به کاربر کمک کنید اما تصمیم نهایی را به او واگذار نمایید.
- در خصوص طراحی و توسعه ای برنامه نیز روش های بهینه وجود دارد که در زیر به آن ها اشاره می کنیم:

- در طراحی اپلیکیشن راندمان و کارایی بهینه در مرتبه ای اول قرار می گیرد - یک اپلیکیشن که به صورت بهینه طراحی شده باشد طبیعتاً با سرعت قابل توجهی اجرا می شود. زمان اجرای (اولیه) اپلیکیشن به طور متوسط نباید بیش از 1 ثانیه به طول بیانجامد. بعلاوه تمامی عملیات طولانی باید به طور ناهمzman اجرا شوند.
- اپلیکیشن خود را طوری طراحی کنید که با سرعت با کاربر تعامل کند - بازخورد می باشد سریعاً در اختیار کاربر قرار گیرد. در صورت اجرای عملیات طولانی بهتر است پیغام کوتاهی در UI برای کاربر به نمایش بگذارد.
- در طراحی اپلیکیشن حداقل مصرف باتری را در نظر بگیرید - اپلیکیشن شما می باشد حداقل میزان مصرف باتری را داشته باشد. در صورتی که اپلیکیشن در UI فعال و قابل مشاهده نیست، تمامی آپدیت های مربوط به رابط کاربری و گوش فراخوان ها (listener) به رخدادها) را غیرفعال نمایید.

شارژر گوش داده و به محض اتفاق افتادن این رخداد، آپدیت های طولانی را فعال نمایید. چنانچه لازم است داده های حجمی را از طریق آپدیت از سرور خارجی واکشی نمایید، بهترین گزینه سرویس Google push notification می باشد. با پیاده سازی این سرویس شما تنها زمانی اجازه ای اتصال به اینترنت را می دهید که داده های لازم در دسترس باشند.

- در دسترس قرار دادن داده های اخیر در اختیار کاربر در زمان راه اندازی اولیه اپلیکیشن - اپلیکیشنی که شما می نویسید می بایست در صورت امکان داده های اخیر را به محض بالا آمدن برنامه در اختیار کاربر خود قرار دهد. بنابراین توصیه می شود برای واکشی داده ها از سرویس دهنده های خارجی (external servers) از service ها بهره بگیرید تا از این طریق عملیات بازیابی اطلاعات و لایه های UI اپلیکیشن از هم جدا شوند.

- بی مورد به اینترنت خودداری کند (push notification) یک سرویس است که ارتباط بین سرویس دهنده و سرویس گیرنده را فراهم می آورد. موارد کاربرد آن عبارت است: همگام سازی، اعمال تغییرات بلادرنگ بر روی سرویس گیرنده، چت سرویس دهنده. به عبارت دیگر push notification پیامی است که به کاربر خارج از اپلیکیشن ارائه می دهید. همچنین توصیه می شود وضعیت جاری اتصال به اینترنت را بررسی نمایید. زمانی که دستگاه به wifi دسترسی دارد، طبیعتاً اپلیکیشن شما امكان دانلود اطلاعات بیشتری را خواهد داشت.

برای کسب اطلاعات بیشتر در خصوص اصول طراحی اپلیکیشن های اندرویدی، می توانید به آدرس <http://developer.android.com/design/index.html> مراجعه نمایید.

**نکته:** در خصوص انتخاب اسم (filename) برای آیکون ها چند نکته لازم به توضیح است: 1. نباید در اسم آیکون ها از حروف و کاراکترهای خاص استفاده نمایید 2. اسم آیکون نباید با عدد آغاز شود 3. لازم است اسم آیکون با حروف کوچک نوشته شود.

ساخت	محتوای	تبلیغاتی:
<a href="http://developer.android.com/distribute/tools/promote/device-art.html">http://developer.android.com/distribute/tools/promote/device-art.html</a>	- سایت	به شما این امكان را می دهد تا محتوای تبلیغاتی برای اپلیکیشن خود

ایجاد نمایید و عکس برنامه‌ی خود را در یک قاب زیبا جایگذاری نمایید. کافی است یک تصویر از اپلیکیشن در صفحه نمایش تهیه نموده و سپس آن را بر روی دستگاه مناسب جایگذاری نمایید.

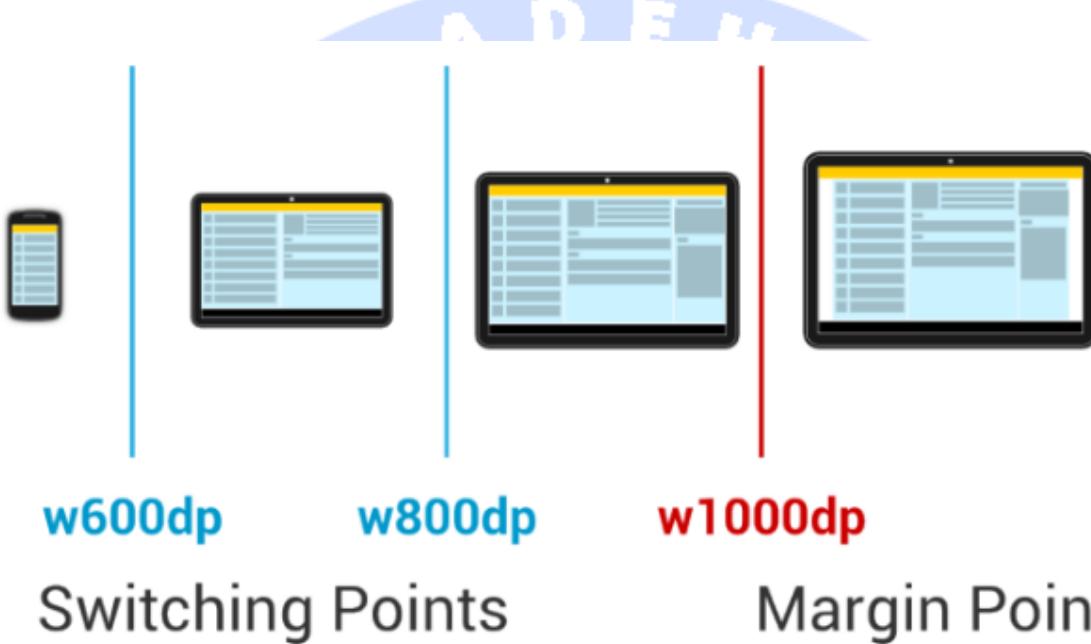


## ۱۰-۵- طراحی انعطاف پذیر و واکنش گرا برای اپلیکیشن (Responsive design)

امروزه قابلیت بزرگ یا کوچک شدن اپلیکیشن متناسب با نمایشگر و عرض صفحه‌ی دستگاه میزبان یک ویژگی پایه‌ای قملداد می‌شود که تقریباً هر برنامه‌ای باید از آن برخوردار باشد. از این رو توصیه می‌شود حتماً اپلیکیشن خود را طوری طراحی نمایید که با توجه به نمایشگر دستگاه خود را اندازه بندی کرده و ظاهری زیبا را در راستای تجربه‌ی کاربری مطلوب از خود ارائه دهد.

در یک نمایشگر کوچک، اپلیکیشن می بایست در لحظه تنها یک fragment را به نمایش بگذارد. در حالی که در صورت عریض بودن دستگاه میزبان بایستی بتواند دو یا حتی طور سه را در آن واحد نمایش دهد.

در تصویر زیر این رویکرد در طراحی لایه ی رابط کاربری اپلیکیشن به راحتی مشاهده می شود.



چنانچه عرض صفحه نمایش دستگاه میزبان از یک حد معین بزرگتر می شود، بهتر است برای ارائه ی نوشته ها در UI از حالت نمایش تمام صفحه استفاده نکنید. این حد معمولاً بالای w1000dp می باشد. تحقیقات انجام گرفته حاکی از آن است که در این حالت کاربر مجبور می شود جهت خواندن محتوا سر خود را بیش از حد به چپ یا راست حرکت دهد که در کل تجربه ی کاربری ضعیفی را به دنبال خواهد داشت.

یک راه برای پیاده سازی margin points این است که به فایل res/values/dimens.xml دسترسی پیدا کرده و برای خطوط حاشیه (margin) ها) مقدار یا اندازه ی معینی را تعریف نمایید. پس از آن می توانید با استفاده از resource qualifier فایل مربوطه، برای نمایشگرهای عریض تر مقادیر متفاوت تعیین نمایید.

## 11-5-استفاده از theme و style در اپلیکیشن

### 11-5-طراحی ساده و بھینه برای اپلیکیشن های اندرویدی با material design/تم گذاری و material design

اصول و رہنمودهای طراحی UI برای اپلیکیشن های اندرویدی طی سالیان تغییر زیادی کرده است. اولین تغییر بزرگ در طراحی ظاهر اپلیکیشن های اندرویدی با ویرایش 3.0 اندروید و تحت نام Holo style معرفی شد. از ویرایش 5.0 سیستم عامل اندروید، طراحی UI بار دیگر با ارئه ای material design (طراحی ساده و تا حد امکان بدون عکس) متحول شد. این الگوی طراحی رابط کاربری، مفهوم depth را در layout معرفی نموده و جهت تعامل با کاربر بیشتر از انیمیشن بهره می گیرد.

آدرس زیر تعداد زیادی منبع قابل دانلود همچون مجموعه آیکون ها را در اختیار توسعه دهندگان قرار می دهد: <https://developer.android.com/design/downloads/index.html>

از ویرایش 5.0 اندروید (API 21)، الگوی material design برای طراحی ظاهر اپلیکیشن توصیه می شود.

Material design یک الگو جهت طراحی UI است که هدف اصلی آن ادغام اصول قدیمی طراحی با قابلیت ها و فناوری های نوین می باشد. این راهنمای جامع همچنین توانسته تعادلی میان کارایی و زیبایی را به ارمغان بیاورد و سازوکاری جالب برای طراحی سایت های جدید ارائه دهد. محیط و بستر اجرای اندروید (android platform) تم نوین، المان های رابط کاربری (widget) و API جدید برای سایه انداختن (shadow) های اختصاصی) و پیاده سازی انیمیشن در اختیار توسعه دهندگان قرار می دهد.

Material design با تخصیص elevation level (میزان فاصله) بین دو المان که بر روی یکدیگر قرار می گیرند) به view ها، این امکان را برای طراح مهیا می کند تا المان های رابط کاربری را بر روی یکدیگر ترسیم نماید. Elevation level را باید بر حسب dp به view تخصیص دهید. حال جهت تنظیم میزان فاصله ای دو المان که بر روی هم قرار می گیرند، می بایست خصیصه layout را در تعریف فایل android:elevation (attribute) نمایید.

جهت مقداردهی میزان elevation بین دو المان با کدنویسی (در کلاس های activity)، لازم است متند View.setElevation() را فراخوانی نمایید. اندروید بر اساس مقدار اختصاص یافته به سایه های اختصاصی (shadow) بر روی المان ها می اندازد.

همچنین API ها و توابع بهینه تری را برای پیاده سازی اینیمیشن ارائه نموده و همچنین تنوع اینیمیشن ها را نسبت به قبل افزایش داده است.

## 2-11-5-شرح مفهوم theme و style

اندروید به شما این امکان را می دهد تا ظاهر کامپوننت های اندرویدی خود (همچون رنگ و فونت) را در فایل های XML تعریف نمایید. این امکان به شما اجازه می دهد تا تمامی attribute هایی که ظاهر اپلیکیشن را تعریف و سبک دهی می کنند به صورت یکجا و در یک مکان واحد مقداردهی نمایید.

آن دسته از المان هایی که در فایل منابع و محتوا (resource) برای سبک دهی به (تنظیم و طراحی ظاهر) view ها مورد استفاده قرار می گیرند، style نامیده می شوند و آن دسته از المان هایی که ویژه ی سبک دهی و تنظیم ظاهر یک activity یا اپلیکیشن تعییه شده باشند، theme نام دارند.

به منظور تعریف theme یا style در پوشه ی اصلی res/values/پروژه خود ذخیره نمایید. گره اول و اصلی (root node) فایل XML مورد نظر می باشد تگ <resources> می باشد. به منظور تعریف style و theme، می باشد تگ style و یک خصیصه ی name (attribute) تعییف نمایید. این المان می تواند یک چندین آیتم در دل خود داشته باشد و مقادیر attribute های نام گذاری شده را تعیین نماید.

کد زیر نمونه ای از تعریف یک style را نمایش می دهد.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="text">
        <item name="android:padding">4dip</item>
```

```

<item name="android:textAppearance">?android:attr/textAppearanceLarge</item>
<item name="android:textColor">#000000</item>
</style>
<style name="layout">
    <item name="android:background">#C0C0C0</item>
</style>
</resources>

```

به منظور تعریف style برای المان های موجود در فایل layout، می بایست به این صورت اقدام نمایید:

style ها و theme ها می توانند به واسطه ی attribute پدر (میزبان) تگ style، ویژگی های والد خود را به ارث برد و از inheritance پشتیبانی نمایند. در واقع بدین وسیله style مورد نظر تمامی تنظیمات را از style پدر خود به ارث برد و می تواند attribute های انتخابی را بازنویسی کند. (overwrite)

### 12-5-دسترسی و اشاره به theme ها در attribute جاری

اندروید تمامی attribute های استاندارد که امکان style دهی آن ها وجود دارد را در فایلی به نام R.attr کرده است. این فایل تحت آدرس <http://developer.android.com/reference/android/R.attr.html> قابل دسترسی می باشد.

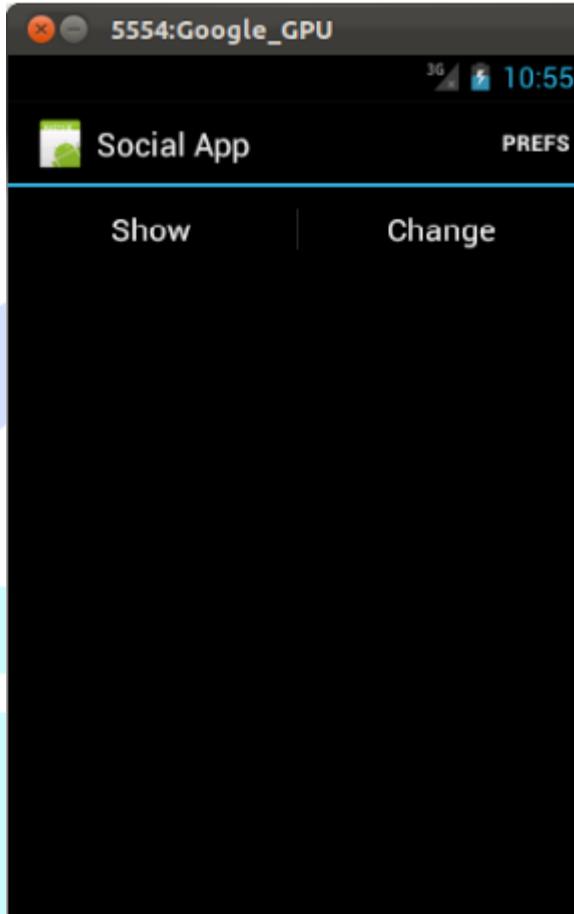
می توانید با استفاده از دستور نگارشی (notation) ?android:attr به theme attribute دیگر شما می توانید با استفاده از این دستور به خصیصه اندروید دسترسی داشته باشید. به عبارت دیگر شما می توانید با استفاده از این دستور به خصیصه (attribute) در theme style اشاره کنید.

به عنوان مثال کد ?android:attr/listPreferredItemHeight به اندروید دستور می دهد تا مقدار اختصاص یافته به خصیصه listPreferredItemHeight را در تم جاری بکار ببرد.

فایل layout زیر به المان های دکمه در UI استایل دکمه های ویرایش 4.0 اندروید را اعمال می کند.

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <LinearLayout
        style="?android:attr/buttonBarStyle"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >
        <Button
            android:id="@+id/Button01"
            style="?android:attr/buttonBarButtonStyle"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Show" />
        <Button
            android:id="@+id/Button02"
            style="?android:attr/buttonBarButtonStyle"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Change" />
    </LinearLayout>
    <EditText
        android:id="@+id/myView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10" >
        <requestFocus />
    </EditText>
</LinearLayout>
```



## Theme-5-12-4 چیست؟

عبارت است یک **style** که به جای یک **view** به کل یک **activity** (یک صفحه از اپلیکیشن) یا اپلیکیشن اعمال می شود. روش تعریف آن با تعریف **style** تفاوتی ندارد.

مثال بعدی نحوه ای تعریف یک **theme** اختصاصی را با ارث بری از **theme** درون ساخته‌ی محیط **اندروید** (**platform**) شرح می دهد.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="MyTheme" parent="android:Theme.Light">
        <item name="android:windowNoTitle">true</item>
        <item name="android:windowBackground">@color/translucent_red</item>
        <item name="android:listViewStyle">@style/MyListView</item>
    </style>
```

```
<style name="MyListView" parent="@android:style/Widget.ListView">
    <item name="android:listSelector">@drawable/ic_menu_home</item>
</style>
</resources>
```

## 13-5-استفاده از theme های خود محیط اندروید

### 13-5-استفاده از material design و کتابخانه‌ی لازم برای پشتیبانی از آن در طراحی اپلیکیشن اندروید

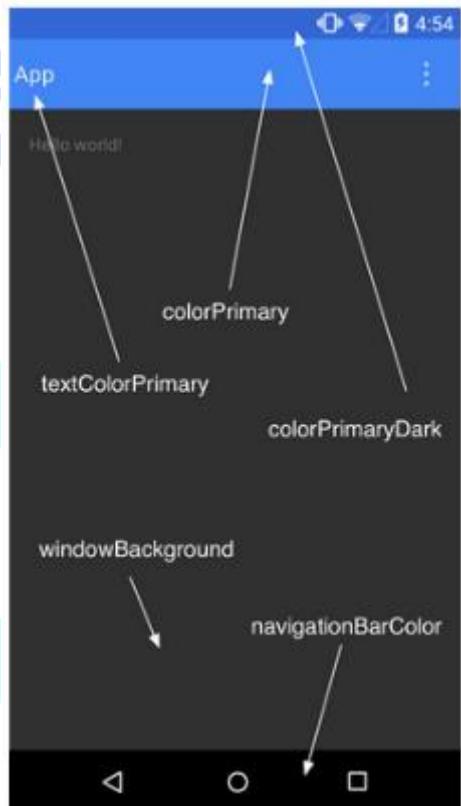
از API/ورژن 21 به بعد کتابخانه‌های اندروید، توسعه دهنده‌گان به استفاده از material design جهت طراحی ظاهر اپلیکیشن ترغیب می‌شوند.

- @android:style/Theme.Material (dark version)
- @android:style/Theme.Material.Light (light version)
- @android:style/Theme.Material.Light.DarkActionBar

برای پیاده سازی material design در طراحی اپلیکیشن‌هایی که برای ویرایش‌های قدیمی اندروید نوشته شده‌اند، بایستی از یک Support library استفاده نمایید که theme و style های این الگوی طراحی UI را در اختیار شما قرار دهد. برای دسترسی به این کتابخانه می‌توانید به آدرس مراجعه فرمایید. با استفاده از این کتابخانه می‌توانید از material design در طراحی ظاهر برنامه در ورژن‌های قدیمی تر اندروید بهره بگیرید.

## 13-5- تنظیم رنگ های پایه‌ی theme (Styling the color palette)

با معرفی material design شما می‌توانید رنگ های پایه‌ی تم را سفارشی تنظیم نمایید.  
تصویر زیر تعداد زیادی از این رنگ‌ها را نمایش می‌دهد.



کد زیر نحوه‌ی استایل دهنده و تنظیم رنگ تصویر بالا در فایل XML را نمایش می‌دهد (این فایل در پوشه‌ی /res/values/ تعریف شده است).

```
<resources>
<style name="AppTheme" parent="android:Theme.Material">
    <!-- Main theme colors -->
    <!-- your app branding color for the app bar -->
    <item name="android:colorPrimary">@color/primary</item>
    <!-- darker variant for the status bar and contextual app bars -->
    <item name="android:colorPrimaryDark">@color/primary_dark</item>
    <!-- theme UI controls like checkboxes and text fields -->
    <item name="android:colorAccent">@color/secondary</item>
</style>
</resources>
```

می توانید با مقداردهی `android:statusBarColor` ظاهر `status bar` (نوار نمایشگر وضعیت کلی دستگاه) را نیز تنظیم و به اصطلاح سبک دهی نمایید. به صورت پیش فرض، `android:colorPrimaryDark` مقدار `android:statusBarColor`

### 3-13-5-سبک دهی و تنظیم ظاهر view های فردی و view group ها

از ویرایش 5.0 اندروید این امکان برای برنامه نویس فراهم شده تا `android:theme` را برای یک `view` تنظیم و مقداردهی کند. با استفاده از این `attribute` می توان `theme` و ظاهر کلی یک `view` و `view` های فرزند آن را تغییر داد.

### تمرین: تعریف theme ها و استفاده از آن ها

#### هدف تمرین

در تمرین حاضر علاوه بر استفاده از `theme` های پیش فرض خود اندروید، `theme` های اختصاصی و دلخواه خود را تعریف خواهید نمود. می توانید برای این تمرین از یک اپلیکیشن آماده استفاده کنید و یا با استفاده از ویزارد ساخت پروژه محیط برنامه نویسی اندروید (IDE) یک پروژه جدید تعریف نمایید.

### 4-14-استفاده از پیش تعریف شده و آماده

اپلیکیشن را جهت استفاده از `android:Theme.Material.Light.DarkActionBar` تنظیم نموده

و آن را تست نمایید. سپس یک `theme` دیگر ایجاد کرده و آن را نیز تست کنید.

می توانید `theme` دلخواه خود را انتخاب نمایید.

### تمرین: پیاده سازی یک theme اختصاصی

نکته: تیم توسعه دهندگان اندروید در Google قالب های آماده و الگوی های ساخت پروژه (template) را به طور مداوم تغییر می دهند. به دنبال این تغییر ممکن است فایل های سبک دهی (style resource) از قبل ایجاد شده باشند یا `theme` های پایه که تم های دیگر از آن ارث می برند (extend می شوند) به کلی تغییر کرده باشد.

ابتدا فایل styles.xml را برای ورژن اندرویدی که برنامه را بر روی آن تست می کنید، ایجاد نمایید. برای این منظور لازم است بخش qualifier ورژن (version qualifier) مربوطه را برای پوشش ای که دربردارنده ی فایل مورد نظر است، ارائه نمایید.

یک ثابت رنگ به نام my-color اضافه نموده که مقدار رنگ #b0b0ff را در فایل تعریف می کند.

(property) را ایجاد نمایید. با اضافه شدن این style می بايست مقدار خصوصیت AppTheme (property) را ایجاد نمایید. با مقدار رنگی که شما تعریف می کنید جایگزین شود.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="my_color">#b0b0ff</color>
    <style name="AppTheme" parent="@android:style/Theme.Material.Light.DarkActionBar">
        <item name="android:windowBackground">@color/my_color</item>
    </style>
    <style name="ToolbarStyling">
        <item name="android:background">@color/colorPrimary</item>
    </style>
</resources>
```

این تم را به اپلیکیشن خود اعمال نمایید. هم اکنون رنگ پس زمینه ی اپلیکیشن بایستی تغییر کرده باشد.



بخش چهارم :

## آموزش ساخت wallpaper/تصاویر زنده در اندروید

آموزش حاضر به شرح نحوه‌ی ساخت تصاویر زنده برای سیستم عامل اندروید می‌پردازد. پروژه‌های این مبحث در محیط برنامه نویسی Eclipse 4.2، با ویرایش 1.6 زبان Java نوشته شده و مبتنی بر ورژن 4.1 سیستم عامل اندروید می‌باشد.

15-5- مرور کلی

## 5-15-1 / تصاویر زنده Live Wallpapers

تصاویر پس زمینه ای پویا و تعاملی هستند که در صفحه ای اصلی سیستم عامل home screen می نشینند. این تصاویر زنده از بسیاری جهات شبیه به دیگر اپلیکیشن های اندرویدی بوده و قادر هستند از اغلب قابلیت های سیستم اندروید مانند سایر برنامه ها بهره ببرند.

### 5-15-2 - نحوه ساخت live wallpaper

به منظور ساخت live wallpaper لازم است یک فایل XML ایجاد نموده و ویژگی های کلی آن را در فایل مزبور اعلان نمایید. فایل نام برده می باشد توصیفی کلی از اپلیکیشن، یک پیش نمایش و لینک به تنظیمات اکتیویتی Activity که در قالب preferences تعریف می شود را دربرداشته باشد. شما بعده ها از طریق preferences می توانید activity را مطابق نیاز تنظیم نمایید.

سپس یک سرویس تعریف می کنید که کلاس WallpaperService را به ارث می برد. در واقع تمامی تصاویر زنده در سیستم اندروید از این کلاس پایه ارث بری کرده و مشتق می شوند. متد onCreateEngine() را پیاده سازی کرده و یک آبجکت از جنس android.service.wallpaper.WallpaperService.Engine نمایید. این آبجکت ها رخدادهای مربوط به چرخه حیات (lifecycle event)، اینیمیشن (متحرک سازی) و ترسیم live wallpaper را مدیریت می کنند. کلاس Engine متدهای مربوط به چرخه حیات را در اختیار توسعه دهنده قرار می دهد که از جمله ای آن می توان به onOffsetsChanged(), onVisibilityChanged(), onSurfaceCreated(), onCreate() و onCommand() onTouchEvent() اشاره کرد.

برای استفاده از سرویس مورد نظر می باشد داخل فایل تنظیمات اپلیکیشن (manifest) مقدار android.permission.BIND\_WALLPAPER را (داخل تگ service) برابر android:permission قرار داده و سپس داخل تگ intent-filter مقدار خصیصه ای android:name را بر android.service.wallpaper.WallpaperService تنظیم نمایید.

همچنین لازم است داخل فایل تنظیمات (AndroidManifest.xml) اعلان نمایید که اپلیکیشن از امکان android.software.live\_wallpaper استفاده می کند. به دنبال این تنظیم، دستگاه هایی که از live wallpaper پشتیبانی نمی کنند، قابلیت نصب نرم افزار شما را نخواهد داشت.

### 5-15-3- استفاده از intent برای تنظیم wallpaper

می توانید یک دکمه تعریف کنید که با کلیک بر روی آن یک intent فعال شده و این intent سبب راه اندازی activity دیگری می شود. سپس در دوم امكان انتخاب Wallpaper را برای کاربر فراهم آورید.

```
// Button to set the Wallpaper
public void onClick(View view) {
    Intent intent = new Intent(WallpaperManager.ACTION_CHANGE_LIVE_WALLPAPER);
    intent.putExtra(WallpaperManager.EXTRA_LIVE_WALLPAPER_COMPONENT,
                  new ComponentName(this, MyWallpaperService.class));
    startActivity(intent);
}
```

### مثالی از پیاده سازی Live Wallpaper در اندروید

یک پروژه جدید به نام de.vogella.android.wallpaper ایجاد نمایید. لازم به ایجاد activity نیست.

پوشه ای /res/xml و فایل mywallpaper.xml نیز را ایجاد نمایید.

```
<?xml version="1.0" encoding="UTF-8"?>
<wallpaper
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:thumbnail="@drawable/icon"
    android:description="@string/wallpaper_description"
    android:settingsActivity="de.vogella.android.wallpaper.MyPreferencesActivity"/>
```

این فایل یک پیش نمایش (در قالب یک تصویر کوچک) و توصیف از wallpaper را شامل می شود. می توانید یک لینک به activity که امکان تنظیم آن را برای کاربر فراهم می کند، در فایل XML لحاظ نمایید. این فایل محتوا به فایل تنظیمات اپلیکیشن (AndroidManifest.xml) متصل بوده و به آن لینک می شود.

با مقداردهی android:thumbnail در فایل حاضر قادر خواهید بود یک تصویر کوچک به عنوان پیش نمایش از live wallpaper فعال ارائه نمایید. مقداری که به این attribute انتساب می دهید در واقع به یک اشاره می کند.

محتوای فایل AndroidManifest.xml به صورت MyWallpaperService را جهت تعریف سرویس زیر ویرایش نمایید. سپس تگ uses-feature را جهت افزودن قابلیت جدید به اپلیکیشن در این فایل اضافه کنید.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="de.vogella.android.wallpaper"
    android:versionCode="1"
    android:versionName="1.0" >
    <application
        android:icon="@drawable/icon"
        android:label="@string/app_name" >
        <service
            android:name="MyWallpaperService"
            android:enabled="true"
            android:label="Wallpaper Example"
            android:permission="android.permission.BIND_WALLPAPER" >
            <intent-filter>
                <action android:name="android.service.wallpaper.WallpaperService" >
                </action>
            </intent-filter>
            <meta-data
                android:name="android.service.wallpaper"
                android:resource="@xml/mywallpaper" >
            </meta-data>
        </service>
        <activity
            android:name=".MyPreferencesActivity"
            android:exported="true"
            android:label="@string/app_name"
            android:theme="@android:style/Theme.Light.WallpaperSettings" >
            </activity>
        <activity
            android:name=".SetWallpaperActivity"
            android:label="@string/app_name"
            android:theme="@android:style/Theme.Light.WallpaperSettings" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
```

```
</application>
<uses-sdk android:minSdkVersion="10" />
<uses-feature
    android:name="android.software.live_wallpaper"
    android:required="true" >
</uses-feature>
</manifest>
```

کلاس MyPoint را تعریف نموده و المان های ترسیم شده را در آن ذخیره نمایید.

```
package de.vogella.android.wallpaper;
public class MyPoint {
    String text;
    private int x;
    private int y;
    public MyPoint(String text, int x, int y) {
        this.text = text;
        this.x = x;
        this.y = y;
    }
}
```

یک activity جدید ایجاد نمایید. سپس فایل prefs.xml را در جهت نگهداری اطلاعات مربوط به تنظیمات در پوشه res/xml ایجاد کنید.

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android">
    <CheckBoxPreference android:key="touch"
        android:title="Enable Touch"></CheckBoxPreference>
    <EditTextPreference android:key="numberOfCircles"
        android:title="Number of Circles"></EditTextPreference>
</PreferenceScreen>
```

یک activity جدید به نام MyPreferencesActivity به همراه کلاس زیر ایجاد نمایید.

```
package de.vogella.android.wallpaper;
import android.os.Bundle;
import android.preference.Preference;
import android.preference.Preference.OnPreferenceChangeListener;
import android.preference.PreferenceActivity;
import android.widget.Toast;
public class MyPreferencesActivity extends PreferenceActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        addPreferencesFromResource(R.xml.prefs);
        // add a validator to the "numberofCircles" preference so that it only
        // accepts numbers
        Preference circlePreference = getPreferenceScreen().findPreference(
```

```

        "numberOfCircles");
    // add the validator           circlePreference.setOnPreferenceChangeListener(numberCheckListener);
}
/**                                                 * Checks that a preference is a valid numerical value
*/
Preference.OnPreferenceChangeListener numberCheckListener = new OnPreferenceChangeListener() {
    @Override
    public boolean onPreferenceChange(Preference preference, Object newValue) {
        // check that the string is an integer
        if (newValue != null && newValue.toString().length() > 0
            && newValue.toString().matches("\\d*")) {
            return true;
        }
        // If now create a message to the user
        Toast.makeText(MyPreferencesActivity.this,
        "Invalid Input",
        Toast.LENGTH_SHORT).show();
        return false;
    }
};

```

بدنه ی کلاس سرویس Wallpaper را به صورت زیر پیاده سازی کنید.

```

package de.vogella.android.wallpaper;
import java.util.ArrayList;
import java.util.List;
import android.content.SharedPreferences;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.os.Handler;
import android.preference.PreferenceManager;
import android.service.wallpaper.WallpaperService;
import android.view.MotionEvent;
import android.view.SurfaceHolder;
public class MyWallpaperService extends WallpaperService {
    @Override
    public Engine onCreateEngine() {
        return new MyWallpaperEngine();
    }
    private class MyWallpaperEngine extends Engine {
        private final Handler handler = new Handler();
        private final Runnable drawRunner = new Runnable() {
            @Override
            public void run() {
                draw();
            }
        };
        private List<MyPoint> circles;
        private Paint paint = new Paint();

```

```

private int width;
int height;
private boolean visible = true;
private int maxNumber;
private boolean touchEnabled;
public MyWallpaperEngine() {
    SharedPreferences prefs = PreferenceManager
        .getSharedPreferences(MyWallpaperService.this);
    maxNumber = Integer
        .valueOf(prefs.getString("numberOfCircles", "4"));
    touchEnabled = prefs.getBoolean("touch", false);
    circles = new ArrayList<MyPoint>();
    paint.setAntiAlias(true);
    paint.setColor(Color.WHITE);
    paint.setStyle(Paint.Style.STROKE);
    paint.setStrokeJoin(Paint.Join.ROUND);
    paint.setStrokeWidth(10f);
    handler.post(drawRunner);
}
@Override
public void onVisibilityChanged(boolean visible) {
    if (visible) {
        handler.post(drawRunner);
    } else {
        handler.removeCallbacks(drawRunner);
    }
}
@Override
public void onSurfaceDestroyed(SurfaceHolder holder) {
    super.onSurfaceDestroyed(holder);
    this.visible = false;
    handler.removeCallbacks(drawRunner);
}
@Override
public void onSurfaceChanged(SurfaceHolder holder, int format,
    int width, int height) {
    this.width = width;
    this.height = height;
    super.onSurfaceChanged(holder, format, width, height);
}
@Override
public void onTouchEvent(MotionEvent event) {
    if (touchEnabled) {
        float x = event.getX();
        float y = event.getY();
        SurfaceHolder holder = getSurfaceHolder();
        Canvas canvas = null;
        try {
            canvas = holder.lockCanvas();
            if (canvas != null) {
                canvas.drawColor(Color.BLACK);
                circles.clear();
            }
        } catch (Exception e) {
        }
    }
}

```

```

        circles.add(new MyPoint(
            String.valueOf(circles.size() + 1), x, y));
    drawCircles(canvas, circles);
}
} finally {
    if (canvas != null)
        holder.unlockCanvasAndPost(canvas);
}
super.onTouchEvent(event);
}
}

private void draw() {
    SurfaceHolder holder = getSurfaceHolder();
    Canvas canvas = null;
    try {
        canvas = holder.lockCanvas();
        if (canvas != null) {
            if (circles.size() >= maxNumber) {
                circles.clear();
            }
            int x = (int) (width * Math.random());
            int y = (int) (height * Math.random());
            circles.add(new MyPoint(String.valueOf(circles.size() + 1),
                x, y));
            drawCircles(canvas, circles);
        }
    } finally {
        if (canvas != null)
            holder.unlockCanvasAndPost(canvas);
    }
    handler.removeCallbacks(drawRunner);
    if (visible) {
        handler.postDelayed(drawRunner, 5000);
    }
}
// Surface view requires that all elements are drawn completely
private void drawCircles(Canvas canvas, List<MyPoint> circles) {
    canvas.drawColor(Color.BLACK);
    for (MyPoint point : circles) {
        canvas.drawCircle(point.x, point.y, 20.0f, paint);
    }
}
}
}

```

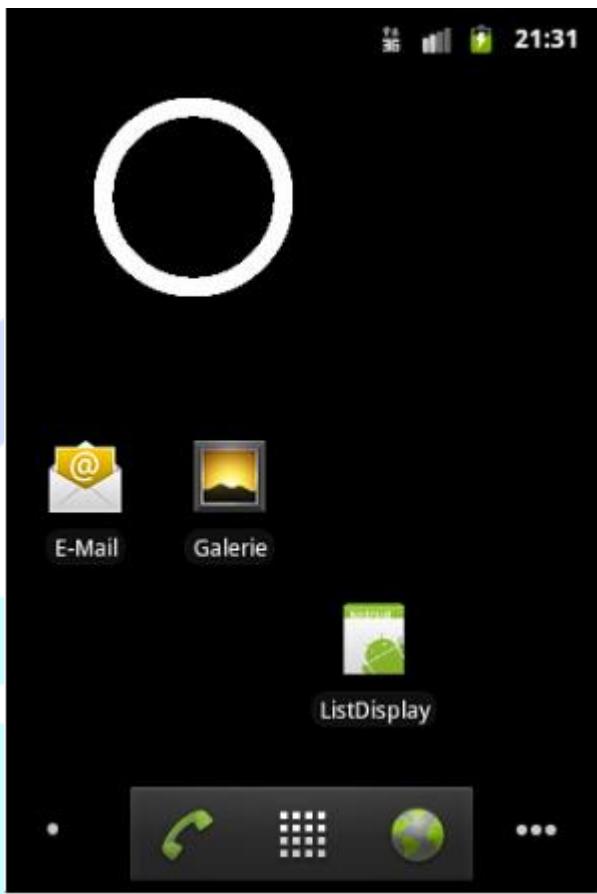
یک کلاس Activity به صورت زیر ایجاد نمایید که ظاهر خود را از یک فایل layout حاوی آبجکت یا المان Button می خواند. این Button از خصوصیت (property) onClick برای اشاره و فراخوانی متده استفاده می کند.

```

package de.vogella.android.wallpaper;
import android.app.Activity;
import android.app.WallpaperManager;
import android.content.ComponentName;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
public class SetWallpaperActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
    public void onClick(View view) {
        Intent intent = new Intent(
WallpaperManager.ACTION_CHANGE_LIVE_WALLPAPER);
        intent.putExtra(WallpaperManager.EXTRA_LIVE_WALLPAPER_COMPONENT,
                      new ComponentName(this, MyWallpaperService.class));
        startActivity(intent);
    }
}

```

پس از راه اندازی، اپلیکیشن بایستی به شما اجازه ی تنظیم wallpaper را بدهد. تصویر پس زمینه مشابه زیر خواهد بود. چنانچه قابلیت Touch را از طریق تنظیمات preferences فعال کرده باشد، در آن صورت می توانید با کلیک بر روی صفحه دایره های جاری را حذف نمایید. علاوه بر آن می توانید تعداد دایره هایی که در UI به نمایش در می آید را تعیین نمایید.



## آموزشگاه تحلیلکر داده ها

**بخش پنجم :**

### رابطWidget های home screen در اندروید/ابزارک ها

#### کاربری صفحه ای اصلی

آموزش حاضر نحوه ایجاد widget ها در صفحه ای اصلی دستگاه اندروید را شرح می دهد.

## 16-5-شرح مفهوم Widget در اندروید

### 16-5-مرواری بر AppWidgets

Widget ها اپلیکیشن های کوچک هستند و به راحتی بر روی یک میزبان که اغلب home screen یا (صفحه ای قفل نمایشگر) دستگاه اندروید هست، قابل جایگذاری می باشد. می توانید به widget ها به دید یک نمونه ای کوچک از کل اپلیکیشن و قابلیت های آن که از طریق صفحه ای اصلی دستگاه قابل دسترسی می باشد، نگاه کرد.

یک widget به عنوان بخشی از فرایند میزبان خود اجرا می شود. این امر لازمه ای آن است که widget از مجوزهای اپلیکیشن میزبان خود برخوردار بوده و آن ها را حفظ کنند.

Widget برای ساخت ظاهر و UI خود در صفحه اصلی دستگاه از کلاس RemoteViews استفاده می کند. کلاس RemoteView این قابلیت را دارد که با همان مجوزهای اپلیکیشن اصلی توسط فرایند دیگر راه اندازی شود. در واقع Widget ها با بهره گیری از این کلاس قادرند با مجوزهای اپلیکیشن اصلی که ابزارک ها به آن متصل هستند، اجرا شوند.

Widget ها برای ساخت ظاهر (UI) خود از broadcast receiver نیز کمک می گیرند. در حقیقت یک آبجکت از جنس RemoteViews را با محتوای layout مربوطه پر می کند (آن را در این آبجکت inflate می کند). آبجکت نام بده سپس به اندروید تحويل داده شده و از آنجا در قالب widget مستقر در صفحه ای اصلی (home screen) برای کاربر به نمایش در می آید.

### 16-5-مراحل ساخت یک widget

1. یک فایل layout تعریف نمایید.
2. یک فایل XML (AppWidgetProviderInfo) ایجاد نموده که property ها و خصوصیت widget های همچون اندازه، زمان و دفعات تکرار بروز رسانی اطلاعات را مشخص کند.

3. یک BroadcastReceiver مورد widget ایجاد کنید که در ساخت رابط کاربری و ظاهر استفاده قرار می‌گیرد.

4. تنظیمات و کانفیگ Widget را در فایل AndroidManifest.xml درج نمایید.

5. در صورت تمایل می‌توانید یک activity جهت انجام تنظیمات (configuration activity) widget host که به مجرد اضافه شدن نمونه‌ای از widget به میزبان (host) همچون صفحه‌ی اصلی، صداخورده می‌شود.

### widget 5-16-3

قبل از ویرایش 3.1 اندروید، یک widget همیشه تعداد مشخصی خانه (cell) را در صفحه‌ی اصلی دستگاه به خود اختصاص می‌داد. هر خانه معمولاً به اندازه‌ی یک آیکون فضای فراهم می‌کند. برای محاسبه‌ی میزان فضای مورد نیاز یک widget می‌بایست از فرمول رو به رو استفاده کنید:

$$2 \times (\text{device width} / \text{Number of columns}) \times 74$$
 - 2  
برای اجتناب از خطاهای مربوط به گرد کردن لحاظ شده است.

از ویرایش 3.1 به بعد اندروید، کاربران این اجازه را دارند که اندازه‌ی widget را مطابق نیاز تنظیم نمایند. جهت فعال سازی این قابلیت، می‌توانید مقدار android:resizeMode را در فایل تنظیمات XML برای این widget برابر "horizontal|vertical" قرار دهید.

### widget 17-5

#### widget 17-5-1

به منظور تعریف intent filter، لازم است یک broadcast receiver با این ایجاد که المان آن بر روی action

android.appwidget.action.APPWIDGET\_UPDATE تنظیم شده باشد، ایجاد نمایید.

```

<receiver
    android:icon="@drawable/icon"
    android:label="Example Widget"
    android:name="MyWidgetProvider" >
    <intent-filter >
        <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
    </intent-filter>
    <meta-data
        android:name="android.appwidget.provider"
        android:resource="@xml/widget_info" />
</receiver>

```

می توان به receiver یک label و آیکون تخصیص داد. این دو در لیست widget های موجود در launcher اندروید و به عنوان فایلی که با کلیک بر روی آن اپلیکیشن اجرا می شود، لیست می گردد.

می توانید با مقداردهی widget، برای android:name="android.appwidget.provider" مورد نظر meta-data تعريف نمایید. فایل تنظیماتی که آن اشاره می کند، تمامی تنظیمات و کانفیگ widget مورد نظر را شامل می شود. این تنظیمات می تواند مربوط به تناب و دفعات بروز رسانی UI، اندازه و layout (ظاهر، چیدمان کلی) اولیه ی widget باشد.

```

<?xml version="1.0" encoding="utf-8"?>
<appwidget-provider xmlns:android="http://schemas.android.com/apk/res/android"
    android:initialLayout="@layout/widget_layout"
    android:minHeight="72dp"
    android:minWidth="146dp"
    android:updatePeriodMillis="1800000" >
</appwidget-provider>

```

## View-5-17-2 layout های موجود و قابل استفاده

Widget در استفاده از کلاس های View با محدودیت موواجه است. برای تنظیم چیدمان و طرح کلی (layout) می توانید از RelativeLayout و LinearLayout ،FrameLayout به عنوان view ها می توانید از ImageView ،Chromometer ،Button ،AnalogClock استفاده کنید. TextView و ProgressBar

از ویرایش 3.0 اندروید به بعد view های بیشتری در اختیار توسعه دهنده قرار گرفته است که از میان آن ها می توان از AdapterViewFlipper، ListView، GridView، StackView و AdapterViewFlipper نام برد.

برای استفاده از این adapter view ها (کلاس های مشتق شده از AdapterView) شما ملزم به تعریف یک collection view widget هستید.

تنها پل ارتباطی و وسیله‌ی تعامل با view های یک widget از طریق event یا رخداد می باشد. این event را می توان در widget ثبت نموده و به محض تعامل کاربر با widget، اتفاق افتادن آن را اعلان کرد.



کلاس AppWidgetProvider متد onReceive() را پیاده سازی کرده، اطلاعات لازم را استخراج می کند و در نهایت متدهای مدیریت چرخه‌ی حیات widget را صدا می زند.

از آنجایی که می توانید چندین نمونه از یک widget را به صفحه‌ی اصلی (home screen) اضافه نمایید، متدهای مربوط به مدیریت چرخه‌ی حیات widget به دو دسته تقسیم می شوند: 1. متدهایی که تنها برای اولین نمونه اضافه/حذف شده فراخوانده می شوند. 2. متدهایی که به ازای هر نمونه از widget صدا خورده می شوند.

متدها	شرح
onEnabled()	زمانی فراخوانی می گردد که نمونه‌ی home screen برای اولین بار به widget اضافه می شود.

onDisabled()	تنها یکبار زمانی که آخرین نمونه <code>widget</code> از صفحه <code>i</code> اصلی حذف می شود، فراخوانی می گردد.
onUpdate()	این متده است که هر بار بروز رسانی <code>widget</code> فراخوانی می شود. متده مذکور شناسه های <code>appWidgetId</code> آن <code>widget</code> هایی که باید با اطلاعات جدید بروز آوری شوند را به عنوان پارامتر ورودی در قالب آرایه دریافت می کند. توجه داشته باشید که این می تواند تمامی نمونه های <code>AppWidget</code> کلاس <code>Provider</code> را شامل شود یا صرفا زیر مجموعه ای از آن را دربرگیرد.
onDeleted()	با فراخوانی این متده نمونه <code>widget</code> از صفحه <code>i</code> اصلی (widget instance) حذف می گردد.

### 5-17-3 و پردازش ناهمزمان Receiver

همان محدودیت هایی را در زمان اجرای برنامه دارد که یک broadcast receiver `Widget` معمولی با آن مواجه می شود. برای مثال می بایست پردازش خود را نهایتا در عرض 5 ثانیه به اتمام برساند.

یک receiver می بایست عملیات طولانی در سرویس انجام داده و سپس `widget` ها را از آن سرویس بروز رسانی کند.

### 5-18-بروز رسانی widget

یک `widget` داده های جدید خود را بر اساس یک جدول زمانی و در فواصل زمانی مشخص دریافت می کند. برای بروز رسانی یک `widget` دو روش وجود دارد: 1. یکی مبتنی بر فایل تنظیمات XML بوده 2. دیگری توسط سرویس `AlarmManager` اندروید صورت می گیرد.

در فایل تنظیمات widget، شما می توانید یک فاصله ی زمانی مشخص تعیین نمایید که بروزرسانی بر اساس آن صورت گیرد. سیستم پس از گذشت این زمان مشخص بیدار شده و broadcast receiver را جهت بروز رسانی widget با داده های جدید فراخوانی می کند. کم ترین فاصله ی زمانی که آپدیت بر اساس آن رخ می دهد، حدودا 1800000 میلی ثانیه معادل 30 دقیقه می باشد. بدین معنی که widget هر 30 دقیقه آپدیت می شود.

این امکان را برای شما فراهم می کند تا در استفاده از منابع صرفه جویی نموده و widget خود را تعداد دفعات بیشتری بروز رسانی نمایید. برای استفاده از این روش، ابتدا یک سرویس تعریف نموده و سپس با استفاده از کلاس AlarmManager آن را طوری زمان بندی می کنید که در فواید زمانی معین اجرا شود. این سرویس متعاقباً widget را به طور مرتب بروز رسانی می کند.

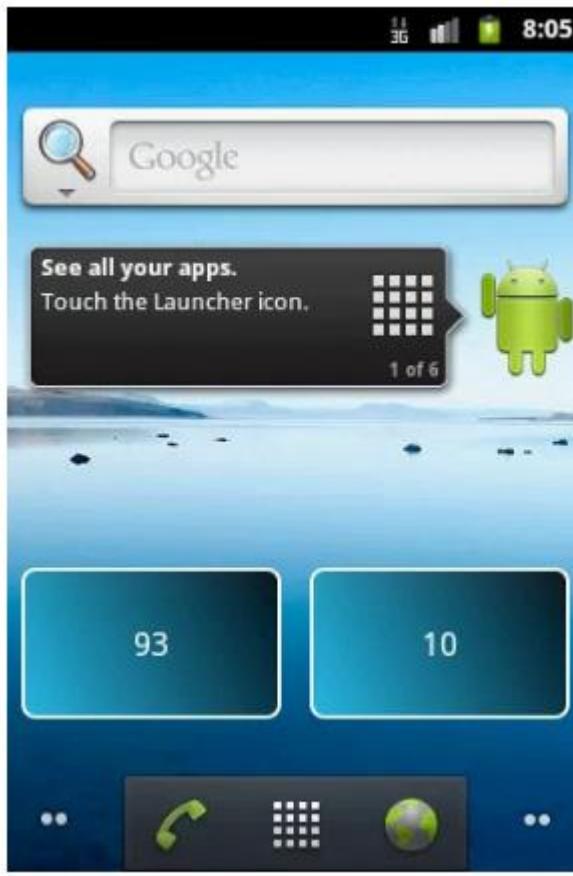
توجه داشته باشید که بروز رسانی در فواید زمانی بیشتر (با تعداد دفعات بروز رسانی بیشتر) ناگذیر سیستم را از حالت صرفه جویی در مصرف باتری خارج ساخته و در نتیجه widget مورد نظر انرژی بیشتری را مصرف خواهد نمود.

## تمرین: پیاده سازی widget و بروز رسانی آن در فواید زمانی معین

### هدف تمرین

در زیر widget ای خواهید ساخت که یک عدد تصادفی را در ال نمایش می دهد. این عدد تصادفی هر 30 دقیقه یکبار بروز رسانی می شود. سپس یک onClickListener ثبت می کنید که با کلیک کاربر بر روی آن، بروز رسانی می شود.

در نهایت ظاهری مشابه زیر خواهید داشت.



## ساخت پروژه و پیاده سازی widget

یک پروژه جدید اندروید به نام de.vogella.android.widget.example ایجاد نموده سپس یک activity جدید در پکیج de.vogella.android.widget.example ایجاد کنید.

یک فایل جدید به نام myshape.xml در پوشه /res/drawable ایجاد نمایید. این فایل در واقع drawable یا فایل تصویری ترسیم شونده ای که به عنوان پس زمینه در widget مورد استفاده قرار می گیرد را تعریف می نماید.

```
<?xml version="1.0" encoding="UTF-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle" >
    <stroke
        android:width="2dp"
        android:color="#FFFFFF" />
    <gradient
```

```

    android:angle="225"
    android:endColor="#DD2ECCFA"
    android:startColor="#DD000000" />
<corners>
    android:bottomLeftRadius="7dp"
    android:bottomRightRadius="7dp"
    android:topLeftRadius="7dp"
    android:topRightRadius="7dp" />
</shape>
```

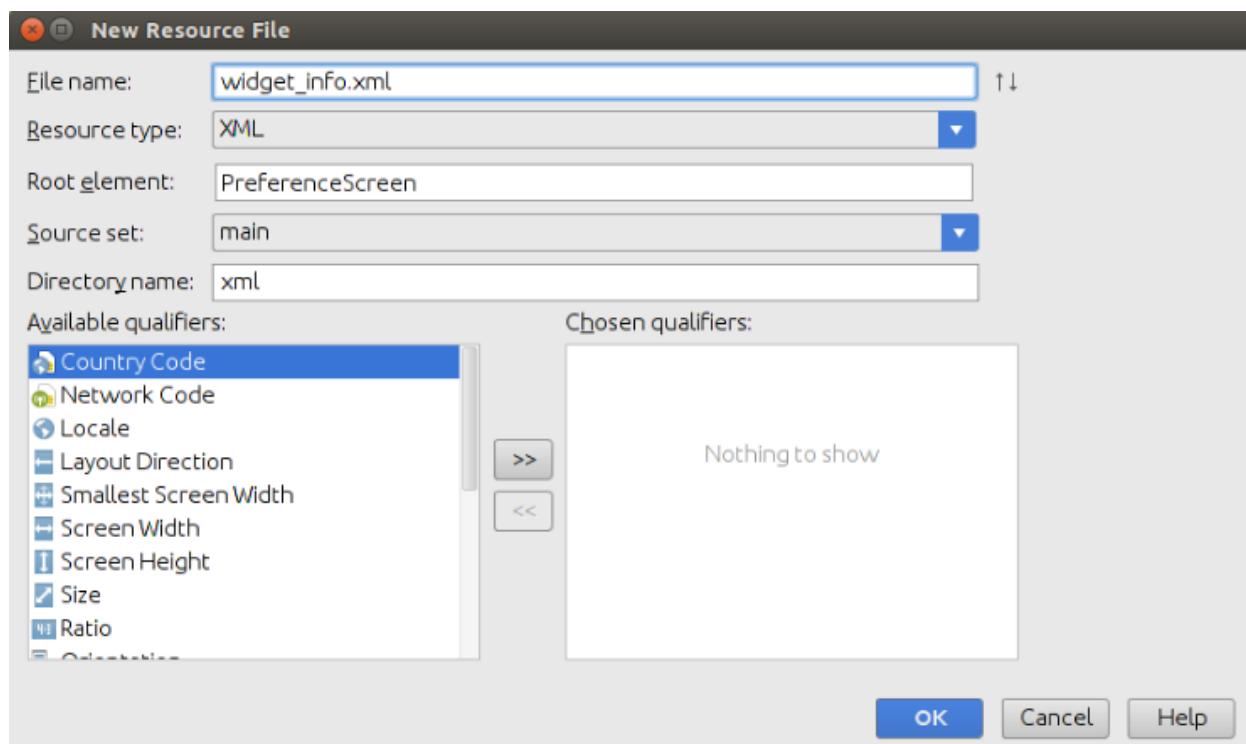
فایل widget\_layout.xml با محتویات زیر را تحت پوشه `res/layout` ایجاد نمایید.

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_margin="8dip"
    android:background="@drawable/myshape" >
    <TextView
        android:id="@+id/update"
        style="@android:style/TextAppearance.Medium"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_gravity="center"
        android:gravity="center_horizontal|center_vertical"
        android:layout_margin="4dip"
        android:text="Static Text" >
    </TextView>
</LinearLayout>
```

یک فایل محتوا (resource) دیگر به نام `widget_info.xml` با کلیک راست بر روی پوشه `res` ایجاد نموده و سپس مسیر رو به رو را طی نمایید:

`.New ▶ Android resource file`



```
<?xml version="1.0" encoding="utf-8"?>
<appwidget-provider xmlns:android="http://schemas.android.com/apk/res/android"
    android:initialLayout="@layout/widget_layout"
    android:minHeight="72dp"
    android:minWidth="300dp"
    android:updatePeriodMillis="300000" >
</appwidget-provider>
```

یک کلاس receiver با پیاده سازی زیر ایجاد نمایید که با هر بار بروز رسانی widget صدا خورده می شود.

```
package de.vogella.android.widget.example;
import java.util.Random;
import android.app.PendingIntent;
import android.appwidget.AppWidgetManager;
import android.appwidget.AppWidgetProvider;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.util.Log;
import android.widget.RemoteViews;
public class MyWidgetProvider extends AppWidgetProvider {
    private static final String ACTION_CLICK = "ACTION_CLICK";
    @Override
    public void onUpdate(Context context, AppWidgetManager appWidgetManager,
        int[] appWidgetIds) {
```

```
// Get all ids
ComponentName thisWidget = new ComponentName(context,
        MyWidgetProvider.class);
int[] allWidgetIds = appWidgetManager.getAppWidgetIds(thisWidget);
for (int widgetId : allWidgetIds) {
    // Create some random data
    int number = (new Random().nextInt(100));
    RemoteViews remoteViews = new RemoteViews(context.getPackageName(),
            R.layout.widget_layout);
    Log.w("WidgetExample", String.valueOf(number));
    // Set the text
    remoteViews.setTextViewText(R.id.update, String.valueOf(number));
    // Register an onClickListener
    Intent intent = new Intent(context, MyWidgetProvider.class);
    intent.setAction(AppWidgetManager.ACTION_APPWIDGET_UPDATE);
    intent.putExtra(AppWidgetManager.EXTRA_APPWIDGET_IDS, appWidgetIds);
    PendingIntent pendingIntent = PendingIntent.getBroadcast(context,
            0, intent, PendingIntent.FLAG_UPDATE_CURRENT);
    remoteViews.setOnClickPendingIntent(R.id.update, pendingIntent);
    appWidgetManager.updateAppWidget(widgetId, remoteViews);
}
```

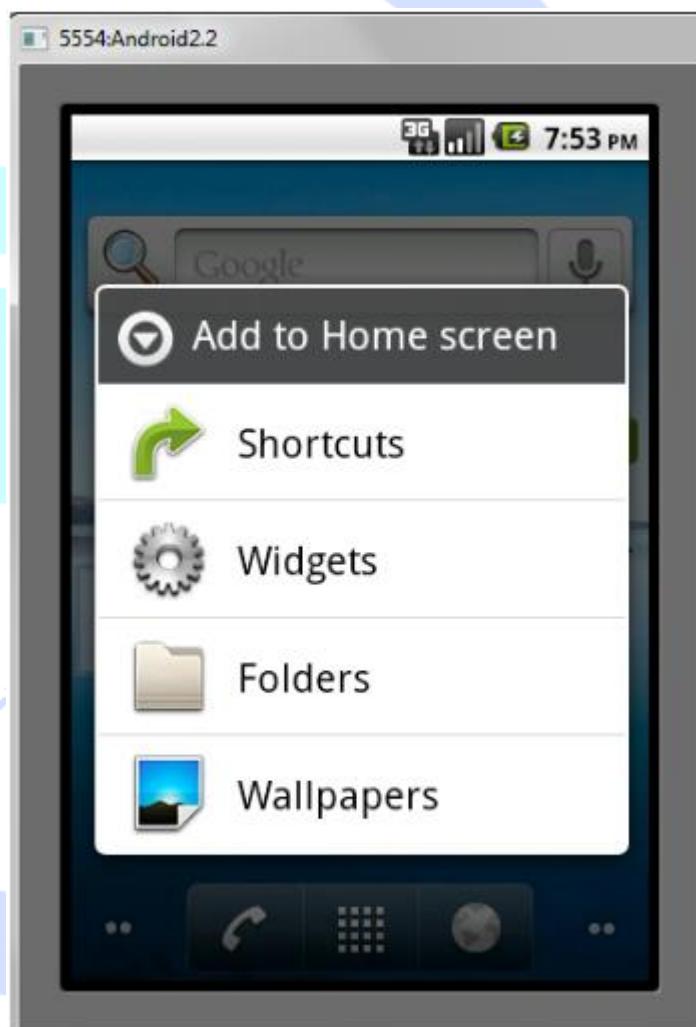
فایل `AndroidManifest.xml` را باز نموده و سپس `widget` خود را به صورت نمایش داده در کد زیر ثبت و تعریف نمایید.

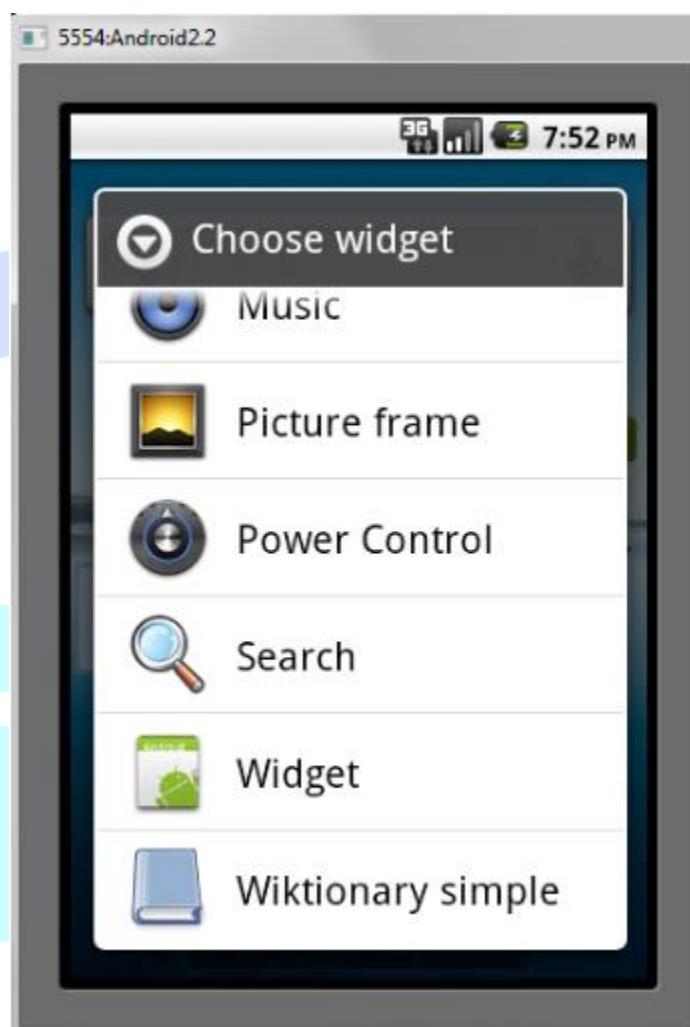
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="de.vogella.android.widget.example"
    android:versionCode="1"
    android:versionName="1.0" >
    <application
        android:icon="@drawable/icon"
        android:label="@string/app_name" >
        <receiver android:name="MyWidgetProvider" >
            <intent-filter >
                <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
            </intent-filter>
            <meta-data
                android:name="android.appwidget.provider"
                android:resource="@xml/widget_info" />
        </receiver>
    </application>
    <uses-sdk android:minSdkVersion="8" />
</manifest>
```

این attribute می اعلان کند که AppWidgetProvider برادرکست را مشخص می کند. ACTION\_APPWIDGET\_UPDATE را پذیرفته و نیز metadata را مربوط به widget کند.

### تست اپلیکیشن

اپلیکیشن خود را بر روی دستگاه اندروید نصب (deploy) نمایید. پس از نصب اپلیکیشن، با استفاده از launcher اندروید widget را بر روی صفحه اصلی دستگاه نصب و آن را تست کنید.





## Collection View Widget-19-5

پیش از هر چیز لازم است درباره **collection widget** توضیح مختصری را در اختیار شما قرار دهیم. **Collection widget** ویژه‌ی نمایش مجموعه‌ای از المان‌های یکسان مورد استفاده قرار می‌گیرد. برای مثال می‌توان به تعدادی عکس از اپلیکیشن **gallery**, تعدادی مقاله از اپلیکیشن **خبرخوان** یا مجموعه‌ای از پیغام/ایمیل‌ها از یک اپلیکیشن ارتباطات سخن گفت. **Collection widget** معمولاً به دو منظور مورد استفاده قرار می‌گیرد: 1. پیمایش در مجموعه‌ای از آیتم‌ها 2. باز کردن آیتمی از مجموعه جهت مشاهده‌ی صفحه‌ی اصلی و جزئیات آن. **Collection widget** دارای نوار اسکرول در کناره‌ی سمت راست بوده که به شما اجازه‌ی پیمایش به صورت عمودی را می‌دهد.

به شما این امکان را می دهد تا از کلاس های `ListView` (جهت پیاده سازی لیست ساده)، `GridView`، `stackview` (لیستی مانند جدول خانه بندی شده) در بستر `widget` استفاده نمایید.

برای پیاده سازی `collection view widget` شما به دو فایل `layout` احتیاج دارید: 1. یکی برای `.widget collection` 2. دیگری برای هر آیتم در `widget`

آیتم های `widget` توسط نمونه ای از `RemoteViewsFactory` ساخته شده و با کلاس `factory` پر می شوند (`Factory class` = در مفهوم شی گرایی، آبجکتی که برای ساخت آبجکت دیگر تعبیه شده در اصطلاح `factory` خوانده می شود).

کلاس `factory` به عنوان یک الگو برای ساخت آبجکت های دیگر ایفا ن نقش می کند. این کلاس در اصل توسط یک سرویس اندروید که خود اعضا و توابع کلاس `RemoteViewsFactory` را به ارت می برد، در اختیار توسعه دهنده قرار می گیرد. لازم به ذکر است که سرویس مزبور برای فعالیت به تنظیم مجوز `android.permission.BIND_REMOTEVIEWS` دارد.

جهت متصل کردن `view` های خود به سرویس، لازم است متده `onUpdate()` را در پیاده سازی `widget` بکار ببرید.

لازم است یک `intent` تعریف نموده که به سرویس مورد نظر دسترسی دارد ( به آن اشاره داشته ) و بعد متده `setRemoteAdapter` را در سطح کلاس `RemoteViews` بکار ببرید.

```
Intent intent = new Intent(context, YourRemoteViewsService.class);
intent.putExtra(AppWidgetManager.EXTRA_APPWIDGET_ID, appWidgetId);
views.setRemoteAdapter(
    appWidgetId,
    R.id.widget_your_id_to_collectionview,
    intent)
```

## 20-فعال سازی یک widget برای قفل نمایشگر دستگاه (lock screen)

از اندروید 4.2 این امکان فراهم شده تا widget های صفحه ای اصلی دستگاه را در صفحه ای قفل (lock screen) نیز جایگذاری نمایید. برای اینکه widget قابلیت قرار گیری در صفحه ای قفل را داشته باشد، لازم است مقدار خصیصه android:widgetCategory (attribute) را در فایل android:widgetCategory قرار دهید. کد زیر نمونه ای را به نمایش می گذارد.

```
<appwidget-provider xmlns:android="http://schemas.android.com/apk/res/android"
    android:widgetCategory="keyguard|home_screen"
    ...
    ...
    ...
</appwidget-provider>
```

در مثال جاری، یک widget تعریف می کنید که قابلیت قرار گیری در صفحه ای اصلی و صفحه ای قفل نمایشگر را داشته باشد. اگر اپلیکیشن خود را هم اکنون recompile و راه اندازی نمایید، خواهید توانست widget را در همین برهه از زمان در صفحه ای قفل نمایشگر قرار دهید.

همچنین می توانید widget category را در زمان اجرای برنامه تشخیص دهید (اینکه آیا زمانی که widget در صفحه ای قفل قرار می گیرد با زمانی که در صفحه ای اصلی نمایش داده می شود دارای ظاهر متفاوتی باشد). برای این منظور، در متدهای AppWidgetProvider.onUpdate() و AppWidgetProvider.onDestroy() شما می توانید با استفاده از کد زیر widget category option یک را بررسی نمایید.

```
Bundle options = appWidgetManager.getAppWidgetOptions(widgetId);
int category = options.getInt(AppWidgetManager.OPTION_APPWIDGET_HOST_CATEGORY, -1);
boolean isLockScreen = category == AppWidgetProviderInfo.WIDGET_CATEGORY_KEYGUARD;
```

با این روش شما می توانید در زمان اجرا تصمیم بگیرید (مشخص کنید) آیا widget ای که اپلیکیشن شما در آن ارائه می دهد بایستی به هنگام نمایش در صفحه ای قفل ظاهری متفاوت داشته باشد یا خیر.

درست مانند زمانی که شما از خصیصه `android:initialLayout` (attribute) جهت تعريف layout اولیه و چیدمان widget های صفحه ای اصلی استفاده می کنید، `widget` را نیز برای تعريف ظاهر `layout` در صفحه ای قفل نمایشگر، داخل فایل `AppWidgetProviderInfo` بکار می برد. این `layout` بلافاصله پس از اینکه `widget` به معنای واقعی مقدار دهی اولیه/راه اندازی می شود با `layout` و ظاهر اصلی جایگزین می گردد.

## تمرین: بروز رسانی widget از طریق یک سرویس

تمرین جاری نحوه ای استفاده از یک سرویس جهت بروز رسانی `widget` با داده های جدید را به نمایش می گذارد.

کلاس زیر را در پروژه خود ایجاد نمایید.

```
package de.vogella.android.widget.example;
import java.util.Random;
import android.app.PendingIntent;
import android.app.Service;
import android.appwidget.AppWidgetManager;
import android.content.ComponentName;
import android.content.Intent;
import android.os.IBinder;
import android.util.Log;
import android.widget.RemoteViews;
public class UpdateWidgetService extends Service {
    private static final String LOG = "de.vogella.android.widget.example";
    @Override
    public void onStart(Intent intent, int startId) {
        AppWidgetManager appWidgetManager = AppWidgetManager.getInstance(this
                .getApplicationContext());
        int[] allWidgetIds = intent
                .getIntArrayExtra(AppWidgetManager.EXTRA_APPWIDGET_IDS);
        // ComponentName thisWidget = new ComponentName(getApplicationContext(),
        //         MyWidgetProvider.class);
        // int[] allWidgetIds2 = appWidgetManager.getAppWidgetIds(thisWidget);
        for (int widgetId : allWidgetIds) {
            // create some random data
            int number = (new Random().nextInt(100));
            RemoteViews remoteViews = new RemoteViews(this
                    .getApplicationContext().getPackageName(),
                    R.layout.widget_layout);
            Log.w("WidgetExample", String.valueOf(number));
        }
    }
}
```

```

    // Set the text
    remoteViews.setTextViewText(R.id.update,
        "Random: " + String.valueOf(number));
    // Register an onClickListener
    Intent clickIntent = new Intent(this.getApplicationContext(),
        MyWidgetProvider.class);
    clickIntent.setAction(AppWidgetManager.ACTION_APPWIDGET_UPDATE);
    clickIntent.putExtra(AppWidgetManager.EXTRA_APPWIDGET_IDS,
        allWidgetIds);
    PendingIntent pendingIntent = PendingIntent.getBroadcast(
        getApplicationContext(), 0, clickIntent,
        PendingIntent.FLAG_UPDATE_CURRENT);
    remoteViews.setOnClickPendingIntent(R.id.update, pendingIntent);
    appWidgetManager.updateAppWidget(widgetId, remoteViews);
}
stopSelf();
super.onStart(intent, startId);
}
@Override
public IBinder onBind(Intent intent) {
    return null;
}
}

```

این کلاس را به عنوان یک سرویس (با تگ service) در فایل تنظیمات AndroidManifest.xml خود تعریف نمایید.

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

را به صورت زیر ویرایش نمایید. در حال حاضر کد زیر سرویس را ساخته و آن را راه اندازی می کند.

```

package de.vogella.android.widget.example;
import android.appwidget.AppWidgetManager;
import android.appwidget.AppWidgetProvider;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.util.Log;
public class MyWidgetProvider extends AppWidgetProvider {
    private static final String LOG = "de.vogella.android.widget.example";
    @Override
    public void onUpdate(Context context, AppWidgetManager appWidgetManager,
        int[] appWidgetIds) {
        Log.w(LOG, "onUpdate method called");
        // Get all ids
        ComponentName thisWidget = new ComponentName(context,

```

```

        MyWidgetProvider.class);
    int[] allWidgetIds = appWidgetManager.getAppWidgetIds(thisWidget);
    // Build the intent to call the service
    Intent intent = new Intent(context.getApplicationContext(),
        UpdateWidgetService.class);
    intent.putExtra(AppWidgetManager.EXTRA_APPWIDGET_IDS, allWidgetIds);
    // Update the widgets via the service
    context.startService(intent);
}
}

```

پس از فراخوانی، این سرویس تمامی widget ها را بروز رسانی می کند. در واقع می توانید با کلیک بر روی یکی از widget های جاری، تمامی widget ها را یکجا بروز آوری نمایید.



## بخش ششم :

# ساخت و پیاده سازی view های سفارشی و ترکیبی در اندروید

این آموزش به شرح ساخت view های اختصاصی و ترکیبی در اندروید می پردازد.

## View-21-5 های اختصاصی

### View-5-21-1 های پیش فرض محیط اندروید

چارچوب نرم افزاری (framework) اندروید تعدادی view درون ساخته و پیش فرض ارائه می دهد که توسعه دهنده می تواند از آن ها به صورت آماده استفاده نماید. کلاسی که تمامی view ها (همان کنترل ها و المان های رابط کاربری) از آن مشتق و ارث بری می شوند، View می باشد.

View ها موظف هستند خود و تمامی المان های داخل خود (view های فرزند که در ViewGroup با آن مواجه می شوید) را اندازه گیری، طرح بندی (layout) و ترسیم نمایند.

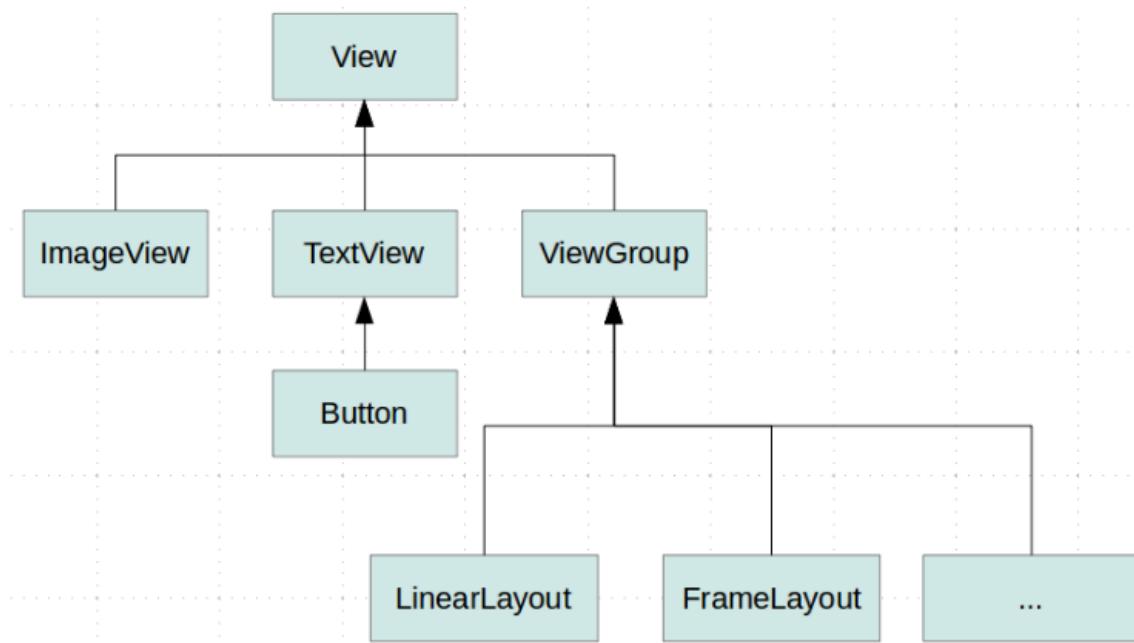
View ها همچنین می بایست اطلاعات مربوط به وضعیت (state) UI را ذخیره کرده و event هایی که با تعامل کاربر با المان های رابط کاربری (لمس نمایشگر) فعال می شوند را مدیریت نمایند.

البته توسعه دهنده این امکان را هم دارند که view های اختصاصی تعریف نموده و آن را در اپلیکیشن خود بکار ببرند.

به منظور تعریف view های سفارشی و دلخواه خود می توانید به یکی از روش های زیر اقدام نمایید:

- (default wiring) - تلفیق view ها با اتصال پیش فرض (Compound view)
- ساخت view های اختصاصی و دلخواه خود (Custom view)
- با ارث بری از یک view ی آماده همچون Button
- با ارث بری از کلاس View

تصویر زیر زنجیره ای ارث بری و سلسله مراتب view های پیش فرض اندروید را به نمایش می گذارد.



View های سفارشی معمولاً با هدف ارائه ی تجربه ی کاربری ویژه و دلخواه که با view های پیش فرض و آماده ی خود اندروید امکان پذیر نیست ساخته می شوند. بعلاوه پیاده سازی view های اختصاصی این امکان را برای توسعه دهنده فراهم می کند تا با ابتکار خود کارایی را افزایش دهد برای مثال در خصوص پیاده سازی layout اختصاصی، برنامه نویس می تواند layout manager را با توجه به نیاز خود تنظیم و بهینه نماید.

## 21-5-اندروید چگونه view hierarchy را ترسیم می نماید!

پس از اینکه انتخاب و مرکز UI بر روی یک activity متمرکز می شود، در همان لحظه بایستی root node (گره یا عنصر اصلی و آغازین) سلسله مراتب layout خود را در اختیار سیستم اندروید قرار دهد. پس از آن سیستم اندروید فرایند ترسیم را آغاز می نماید.

روند ترسیم layout دو مرحله را پشت سر می گذارد:

- پیاده `measure(int, int)` (مرحله ی سنجش و اندازه گیری) – توسط متدهای `measure(int, int)` و `layout(int, int, int)` این مرحله به صورت پیمایش از بالای سلسله مراتب view تا پایین آن اتفاق می افتد. هر view اندازه های خود را ذخیره می کند.

- `layout(int, int, int)` (مرحله ی تنظیم چیدمان و طرح بندی) – این مرحله توسط متدهای `layout(int, int, int)` این مرحله سازی می شود. پیمایش مرحله ی جاری نیز از بالای سلسله مراتب view به پایین آن رخ می دهد. طی این مرحله هر layout manager مسئول چیدمان و موقعیت دهی فرزندان view (های داخل خود) می باشد. لازم به ذکر است که متند فوق برای تنظیم موقعیت view ها از اندازه های بدست آمده در مرحله ی اول استفاده می کند.

**نکته:** مرحله ی `measure` (اندازه گیری) و `layout` (چیدمان) همیشه همزمان اتفاق می افتد.

Layout manager می تواند مرحله ی اندازه گیری را چندین بار اجرا نماید. برای مثال، `LinearLayout` از attribute `weight` پشتیبانی می کند که فضای خالی باقی مانده را بین view ها پخش نموده و `RelativeLayout` تمامی المان یا view های فرزند خود را چندین بار اندازه گیری می کند تا های تعیین شده در فایل `layout constraint` همگی برآورده شوند.

یا activity هر یک می توانند مرحله ی اندازه ی گیری و چیدمان را با فراخوانی متدهای `requestLayout()` راه اندازی نماید.

پس از انجام محاسبات مربوط به اندازه گیری و تنظیم چیدمان المان ها، view ها اقدام به ترسیم می نمایند. جهت راه اندازی این عملیات کافی است متده `invalidate()` از کلاس View خود فراخوانی شود.

### 21-3-استفاده از view های جدید در فایل های layout

View های اختصاصی و ترکیبی (پیچیده) می توانند در فایل های layout تعریف و استفاده شوند. برای نیل به این هدف لازم است اسم view ها را به صورت تمام و کامل در فایل ذکر شده قید نمایید. برای مثال می بایست اسم کلاس و پکیج (پوشه ای اصلی پروژه) را ارائه کنید.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button" />
    <de.vogella.android.ownview.MyDrawView
        android:id="@+id/myDrawView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</LinearLayout>
```

**توجه:** در صورت تمایل می توانید name space خود را در فایل layout اعلان نمایید، مانند name space اندروید.

### 21-4-تهیه ای تصویر آنی (screenshot) از view ها

تمامی کلاس های View این قابلیت را دارند از وضعیت و ظاهر کنونی خود تصویر آنی تهیه نمایند.

```
# Build the Drawing Cache
view.buildDrawingCache();
# Create Bitmap
Bitmap cache = view.getDrawingCache();
```

```
# Save Bitmap
saveBitmap(cache);
view.destroyDrawingCache();
```

## (Compound views) view-22-5

View های ترکیبی (یا کامپوننت های ترکیبی) به View های گفته می شوند که از ترکیب چند View دیگر پدیده آمده باشد.

Compound view ها به شما این امکان را می دهند تا API ها و توابع اختصاصی خود را جهت بروز رسانی و کوئری گرفتن از اطلاعات مربوط به وضعیت view بکار ببرید.

برای این control یک فایل layout تعریف می کنید و آن را به compound view خود تخصیص می دهید. در پیاده سازی compound view، بایستی ارتباط متقابل view ها را تعریف کرده باشید. ابتدا یک فایل layout تعریف می کنید که اعضاء و توابع کلاس ViewGroup مربوطه را به ارث می برد. در این کلاس فایل layout را بارگذاری نموده (inflate) و منطق (کد) اتصال و ارتباط View را پیاده سازی می نمایید.

**نکته:** برای افزایش کارایی و سرعت اجرا، بهتر است view سفارشی که از کلاس View ارث بری کرده ایجاد نمایید. از این طریق می توانید سلسله مراتب view خود و زیرشاخه های آن را به صورت خطی نمایش دهید (flatten view). چرا که در این حالت ترسیم view به پیمایش کمتری نیاز دارد و در صورتی که به شکل درستی پیاده سازی شود، بسیار سریع تر اجرا خواهد شد.

## 23-5-ساخت view های اختصاصی

### 5-پیاده سازی view های اختصاصی

با ارت بری از کلاس View یا یکی از کلاس های مشتق آن (subclass)، شما می توانید view دلخواه خود را ایجاد نمایید.

برای ترسیم view می توانید متدهای onDraw() را بکار ببرید. در این متدهای آبجکت Canvas به عنوان ورودی دریافت می کنید. آبجکت نام برده به شما امکان می دهد تا عملیات ترسیم همچون کشیدن خط، دائرة، درج متن یا bitmap را در سطح آن انجام دهید. در صورتی که view مجدداً ترسیم گردد، شما می توانید متدهای invalidate() را فراخوانی کنید که خود سبب فراخوانی متدهای onDraw() این view می شود.

**نکته:** در صورت تعریف view های اختصاصی، حتماً کلاس ViewConfiguration را بررسی نمایید چرا که این کلاس تعدادی ثابت (constant) درون ساخته دارد که شما می توانید برای تعریف view ها مورد استفاده قرار دهید.

برای ترسیم Views توسعه دهندهان اغلب از 2D Canvas API استفاده می کنند.

### 23-5-اندازه گیری view ها

متدهای onMeasure() از view مورد نظر را صدا می زند. Layout manager سپس پارامترهای layout manager را از layout manager تعیین کند. layout manager از view های داخل خود را بر عهده دارد.

View می بایست متدهای setMeasuredDimension (int, int) را با نتیجه مورد نظر فراخوانی نماید.

### 23-5-تعریف layout manager اختصاصی

برای تعریف layout manager اختصاصی خود می‌توانید اعضا و توابع کلاس ViewGroup را به ارث ببرید. یا انجام کار فوق این امکان برای شما فراهم می‌شود تا layout manger های کارامد و بهینه تری را پیاده سازی نموده یا افکت های دیداری را خلق کنید که در محیط (platform) اندروید وجود ندارد.

یک layout manager سفارشی، به تبع پیاده سازی توابع onLayout() و onMeasure() را بازنویسی (override) نموده و عملیات محاسبه‌ی (اندازه‌ی) المان‌ها فرزند خود را نیز به صورت اختصاصی انجام می‌دهد. برای مثال ممکن است استفاده از ویژگی سنگین و زمان بر layout\_weight از کلاس LinearLayout را کنار بگذارد.

**نکته:** به منظور محاسبه‌ی اندازه‌ی المان محصور (فرزنده) کافی است متده measureChildWithMargins() از کلاس ViewGroup را فراخوانی نمایید.

توصیه می‌شود تمامی پارامترهای اضافی layout را در یک کلاس داخلی درون نمونه‌ی پیاده سازی شده‌ی خود از ViewGroup قرار دهید. برای مثال کلاس LayoutParams به عنوان یک ویژگی در کلاس ViewGroup ویژگی‌های layout, command, parameters و layout\_weight را درون خود کپسوله کرده است و کلاس LinearLayout نیز علاوه بر این ویژگی‌ها، ویژگی layout\_weight را به این ویژگی اضافه نموده است.

### Life Cycle-24-5

#### 24-1 event ها و توابع مربوط به مدیریت چرخه‌ی حیات

یک view زمانی نمایش داده می‌شود که به layout hierarchy متصل باشد. هر view تعداد زیادی تابع hook مدیریت چرخه‌ی حیات نیز خود به تبع به window وصل می‌باشد. دسترسی window به تابع onAttachedToWindow() زمانی فراخوانی می‌شود که پنجره در حافظه بارگذاری شده و در

دارد.

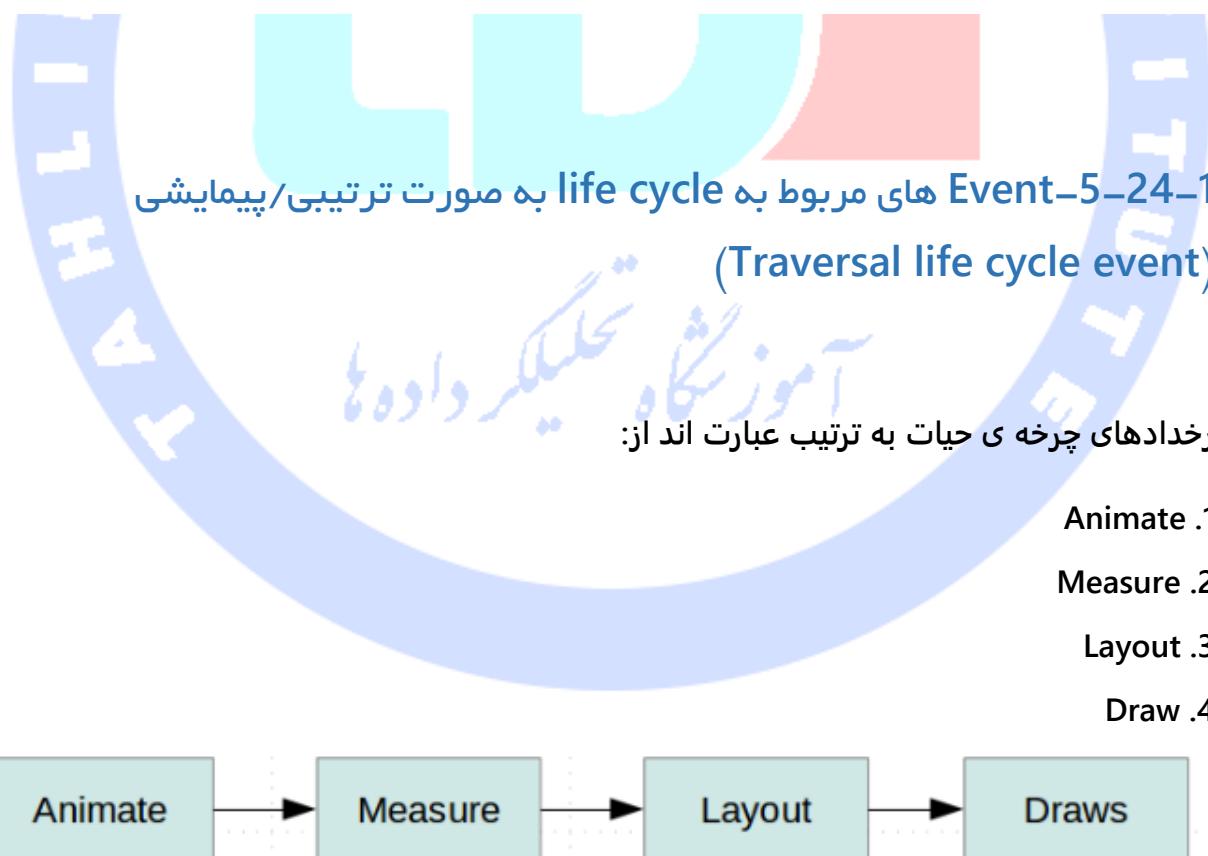
متده onAttachedToWindow() زمانی فراخوانی می‌شود که پنجره در حافظه بارگذاری شده و در دسترس باشد.

متد (parent) زمانی فراخوانی می شود که view از میزبان (parent) خود جدا شده باشد (و البته میزبان نیز خود به یک window وصل باشد). این اتفاق، برای مثال، زمانی رخ می دهد که activity (به عنوان مثال با صدا خورده شدن متده (finished() یا view بازیافت شده باشند.

متد (onAttachedToWindow() زمانی فراخوانی می شود که یک پنجره در دسترس باشد.

متد (onDetachedFromWindow() از والد View پس از حذف استفاده می نماید(حتی اگر به یک پنجره دیگر پیوست باشد) . این رویداد نیز برای Activity به صورت باز پس گیری (حتی وقتی متده (finished() فراخوانی شده باشد) روی می دهد.

متد (onDetachedFromWindow() را می توان جهت متوقف کردن انیمیشن ها و پاک سازی و آزاد نمودن منابع مورد استفاده ای view فراخوانی نمود.



لازم است view ها با نحوه اندازه گیری و چیدمان (layout) خود آشنا باشند. متدهای view اعلان می کند که خود را اندازه گرفته و موقعیت دهنده (layout) نمایند. از آنجایی که این عملیات ممکن است layout دیگر view ها را نیز متاثر کند، متدهای requestLayout() و requestLayout() نیز فراخوانی می شود.

**نکته:** زمانی که چندین layout را داخل هم قرار داده و آن ها را زیاد تودرتونمایید، این امر سبب به اجرا در آمدن الگوریتم فراخوانی بازگشتی می شود. حال زمانی که عمق ساختار درختی یا سلسله مراتب زیاد بوده و در این حین لازم باشد سلسله مراتب مجدداً محاسبه شود، عملیات اندازه گیری و موقعیت دهنده ناگذیر سنگین و طولانی خواهد بود.

متدهای onMeasure() و المان های محصور آن view های فرزند را مشخص می کند. سپس با فراخوانی setMeasureDimension() داخل بدنه ای خود و قبل از اجرای دستور return اندازه ای آن ها را تنظیم می نماید.

متدهای onLayout() تمامی view ها را بر اساس نتیجه یا خروجی متدهای onMeasure() موقعیت دهنده می نماید. این فراخوانی معمولاً تنها یکبار انجام می شود در حالی که onMeasure() می تواند چندین بار صدا خورده شود.

## ۲۴-۵-چرخه ای حیات activity

View ها به event های چرخه ای حیات activity ها دسترسی ندارند. اگر لازم باشد که view ها از رخداد این event ها باخبر شوند، در آن صورت می باشد که view interface در ایجاد نمایید (مربوطه را در بدنه ای کلاس view پیاده سازی کرده) و متدهای مریبوط به مدیریت چرخه ای حیات activity را از طریق آن فراخونی نمایید.

## ۲۵-تعريف attribute های بیشتر برای view های اختصاصی

شما می توانید attribute های اضافی بر سازمان برای view های ترکیبی و اختصاصی خود تعریف نمایید. برای این منظور ابتدا بایستی یک فایل به نام attrs.xml در پوشه res/values خود ایجاد نمایید. در زیر مثالی را می بینید که برای view جدیدی به نام ColorOptionsView تعدادی attribute جدید تعریف شده است.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <declare-styleable name="ColorOptionsView">
        <attr name="titleText" format="string" localization="suggested" />
        <attr name="valueColor" format="color" />
    </declare-styleable>
</resources>
```

جهت استفاده از attribute های نام بردۀ در فایل layout، می بایست آن ها در بخش header فایل XML اعلان نمایید. در کد زیر این کار توسط دستور xmlns:custom صورت گرفته است. مقدار attribute هایی که زیر این دستور درج می شوند، متعاقباً به view مورد نظر نیز اعمال می شوند.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    <!-- define new name space for your attributes -->
    xmlns:custom="http://schemas.android.com/apk/res/com.vogella.android.view.compoundview"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    >
    <!-- Assume that this is your new component. It uses your new attributes -->
    <com.vogella.android.view.compoundview.ColorOptionsView
        android:layout_width="match_parent"
        android:layout_height="?android:attr/listPreferredItemHeight"
        custom:titleText="Background color"
        custom:valueColor="@android:color/holo_green_light"
    />
</LinearLayout>
```

مثال زیر نشان می دهد چگونه کامپوننت های شما می توانند به این attribute ها دسترسی داشته باشند.

```
package com.vogella.android.view.compoundview;
import android.content.Context;
import android.content.res.TypedArray;
import android.util.AttributeSet;
```

```

import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;
public class ColorOptionsView extends View {
    private View mValue;
    private ImageView mImage;
    public ColorOptionsView(Context context, AttributeSet attrs) {
        super(context, attrs);
        TypedArray a = context.obtainStyledAttributes(attrs,
            R.styleable.Options, 0, 0);
        String titleText = a.getString(R.styleable.Options_titleText);
        int valueColor = a.getColor(R.styleable.Options_valueColor,
            android.R.color.holo_blue_light);
        a.recycle();
        // more stuff
    }
}

```

## تمرین: ساخت و پیاده سازی یک view ترکیبی

### ایجاد پروژه

یک پروژه ای جدید اندروید با داده ها و مقادیر زیر ایجاد نمایید.

#### جدول پروژه ای جدید اندرویدی (1,1)

Property	Value
Testing (تست)	Table width
Application Name (اسم اپلیکیشن)	Compound view example
Project Name (اسم پروژه)	com.vogella.android.customview.compoundview
Package name (اسم پوششی حاوی کلاس ها)	com.vogella.android.customview.compoundview
API (Minimum, Target, Compile with)  پایین ترین ویرایش اندروید که برنامه بایستی بر روی آن قابل اجرا باشد، ویرایشی که برنامه برای آن	Latest

**طراحی شده، ورزشی که برنامه با آن  
کامپایل می شود**

Template (قالب آماده و بیان سازی پروژه)	Empty Activity
Activity	MainActivity
Layout	activity_main

## 5-25-1- تعریف و استفاده از attribute های جدید

یک فایل حامل attribute های جدید به نام attr.xml را داخل پوشه‌ی res/values ایجاد نمایید.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <declare-styleable name="Options">
        <attr name="titleText" format="string" localization="suggested" />
        <attr name="valueColor" format="color" />
    </declare-styleable>
</resources>
```

محتوای فایل layout ای که توسط کلاس activity نمایش برای فراخوانی در UI فراخوانی می‌شود را به صورت زیر ویرایش نمایید.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:custom="http://schemas.android.com/apk/res/com.vogella.android.view.compoundview"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:showDividers="middle"
    android:divider="?android:attr/listDivider"
    tools:context=".MainActivity" >
    <com.vogella.android.view.compoundview.ColorOptionsView
        android:id="@+id/view1"
        android:layout_width="match_parent"
        android:layout_height="?android:attr/listPreferredItemHeight"
        android:background="?android:selectableItemBackground"
        android:onClick="onClicked"
        custom:titleText="Background color"
        custom:valueColor="@android:color/holo_green_light"
        />
    <com.vogella.android.view.compoundview.ColorOptionsView
        android:id="@+id/view2"
        android:layout_width="match_parent"
        android:layout_height="?android:attr/listPreferredItemHeight"
        android:background="?android:selectableItemBackground"
```

```

    android:onClick="onClicked"
    custom:titleText="Foreground color"
    custom:valueColor="@android:color/holo_orange_dark"
  />
</LinearLayout>

```

## 5-25-2 ساخت View ترکیبی

فایل زیر به نام view\_color\_options را برای اختصاصی خود ایجاد نمایید.

```

<?xml version="1.0" encoding="utf-8"?>
<merge xmlns:android="http://schemas.android.com/apk/res/android" >
  <TextView
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:layout_centerVertical="true"
    android:layout_marginLeft="16dp"
    android:textSize="18sp"
  />
  <View
    android:layout_width="26dp"
    android:layout_height="26dp"
    android:layout_centerVertical="true"
    android:layout_marginLeft="16dp"
    android:layout_marginRight="16dp"
  />
  <ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginRight="16dp"
    android:layout_centerVertical="true"
    android:visibility="gone"
  />
</merge>

```

اختصاصی خود را به صورت زیر پیاده سازی نمایید.

```

package com.vogella.android.customview.compoundview;
import com.vogella.android.view.compoundview.R;
import android.content.Context;
import android.content.res.TypedArray;
import android.util.AttributeSet;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;
public class ColorOptionsView extends LinearLayout {
  private View mValue;

```

```

private ImageView mImage;
public ColorOptionsView(Context context, AttributeSet attrs) {
    super(context, attrs);
    TypedArray a = context.obtainStyledAttributes(attrs,
        R.styleable.ColorOptionsView, 0, 0);
    String titleText = a.getString(R.styleable.ColorOptionsView_titleText);
    int valueColor = a.getColor(R.styleable.ColorOptionsView_valueColor,
        android.R.color.holo_blue_light);
    a.recycle();
    setOrientation(LinearLayout.HORIZONTAL);
    setGravity(Gravity.CENTER_VERTICAL);
    LayoutInflater inflater = (LayoutInflater) context
        .getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    inflater.inflate(R.layout.view_color_options, this, true);
    TextView title = (TextView) getChildAt(0);
    title.setText(titleText);
    mValue = getChildAt(1);
    mValue.setBackgroundColor(valueColor);
    mImage = (ImageView) getChildAt(2);
}
public ColorOptionsView(Context context) {
    this(context, null);
}
public void setValueColor(int color) {
    mValue.setBackgroundColor(color);
}
public void setImageVisible(boolean visible) {
    mImage.setVisibility(visible ? View.VISIBLE : View.GONE);
}
}

```

### 3-25-5- تنظیم و ویرایش activity

بدنه ی activity را به صورت زیر ویرایش نموده و سپس اپلیکیشن خود را اجرا نمایید.

```

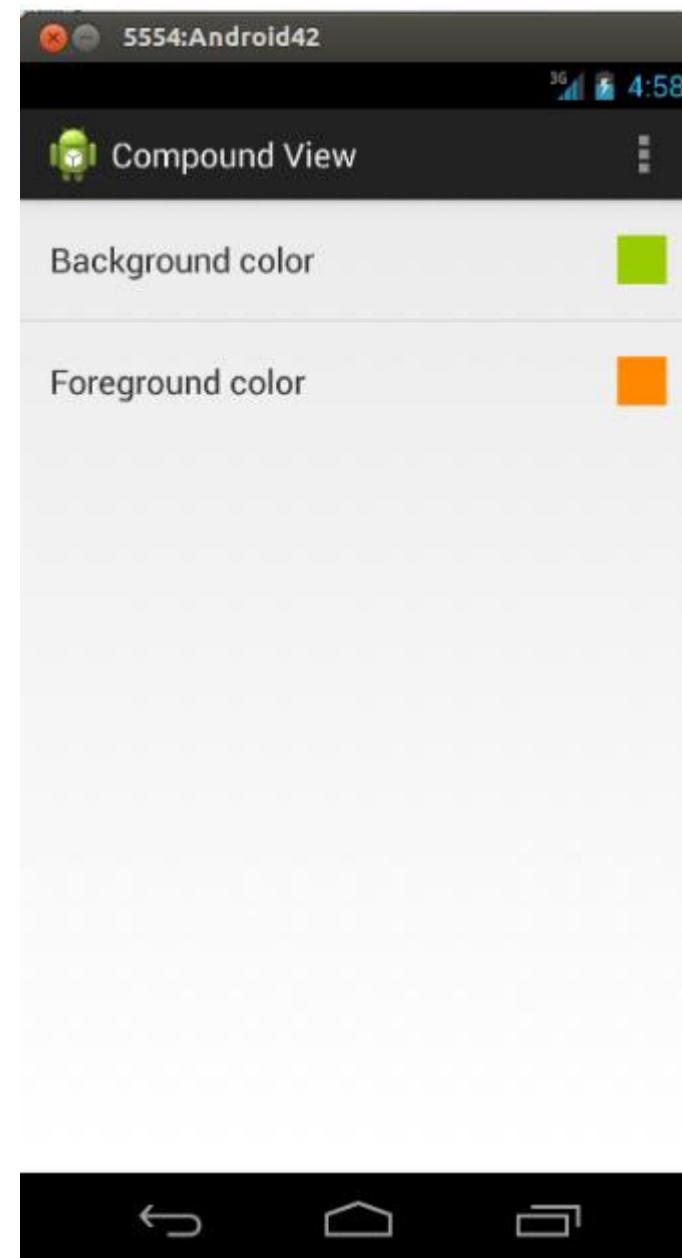
package com.vogella.android.customview.compoundview;
import com.vogella.android.view.compoundview.R;
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.View;
import android.widget.Toast;
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}

```

```
@Override  
public boolean onCreateOptionsMenu(Menu menu) {  
    // Inflate the menu; this adds items to the action bar if it is present.  
    getMenuInflater().inflate(R.menu.activity_main, menu);  
    return true;  
}  
public void onClicked(View view) {  
    String text = view.getId() == R.id.view1 ? "Background" : "Foreground";  
    Toast.makeText(this, text, Toast.LENGTH_SHORT).show();  
}  
}
```

اپلیکیشن در زمان اجرا می بایست ظاهری مشابه زیر داشته باشد.





## (توابع مرتبط با کلاس Canvas API-26-5

### Canvas API-5-26-1 شرح مفهوم

Canvas API به شما اجازه می دهد تا افکت های دیداری و گرافیکی منحصر بفردی خلق نمایید. کلاس Canvas تعدادی تابع جهت ترسیم اشکال در اختیار دارد. برای انجام عملیات ترسیم، توسعه دهنده به 4 مولفه‌ی ابتدایی احتیاج دارد: 1. یک bitmap جهت میزبانی پیکسل‌ها. 2. یک

برای نگه داشتن توابع draw (جهت ترسیم و نوشتن در bitmap) 3. یک کلاس جهت ترسیم مانند 4. یک کلاس paint به منظور تعریف رنگ ها و style های المان های گرافیکی.

در واقع شما بر روی سطح Bitmap اشکال را ترسیم می کنید، توابع ترسیم اشکال را از کلاس Canvas وام گرفته و در نهایت با استفاده از کلاس Paint رنگ و استایل اشکال مورد نظر بر روی سطح canvas را تعیین می نمایید.

## Canvas 26-5-کلاس

آبجکت Canvas آن فایل تصویری پیکسلی یا bitmap ای که اشکال بر روی آن ترسیم می شوند را در برمی گیرد. علاوه بر آن توابعی را ویژه‌ی تنظیم رنگ (drawARGB()), ترسیم عکس پیکسلی (drawBitmap()), درج متن (drawText()), ترسیم مستطیل با گوشه‌های گرد (drawRoundRect()) و غیره ... در اختیار برنامه نویس قرار می دهد.

## Paint 26-5-کلاس

برای تنظیم رنگ، فونت و غیره ... بر روی آبجکت Canvas به یک آبجکت از جنس کلاس Paint نیاز دارید.

کلاس Paint در واقع این قابلیت را در اختیار شما قرار می دهد تا رنگ، فونت و برخی از افکت‌ها را برای انجام عملیات ترسیم مشخص نمایید.

متند () به شما اجازه می دهد تا به سیستم اعلان نمایید آیا تنها outline و خط پیرامون (Paint.Style.FILL) باید ترسیم شود یا تنها بخش رنگ شده (Paint.Style.STROKE) و یا هر دو بخش مذبور.

برای تنظیم کانال آلفا Paint کافی است متند () setAlpha را فراخوانی نمایید.

جهت تنظیم رنگ‌های بیشتر برای آبجکت Paint () می توانید از Shaders استفاده نمایید.

## Shader\_5-26-3

آبجکت shader این امکان را فراهم می کند تا برای آبجکت Paint محتوایی که باید ترسیم شوند را مشخص نمایید. برای مثال، می توانید با استفاده از Shader که کلاس مشتق از BitmapShader هست، یک عکس پیکسلی (Bitmap) تعریف نمایید و سپس در آن شکل هندسی دلخواه را ترسیم نمایید. بدین وسیله شما می توانید به عنوان نمونه، تصویری با گوشه های گرد بکشید. کافی است یک BitmapShader برای آبجکت Paint تعریف نموده و با فراخوانیتابع drawRoundRect()، شکل مسططیل را با گوشه های گرد ترسیم نمایید.

محیط اندروید (platform) تعدادی کلاس مشتق از Shader جهت ترسیم و پیاده سازی شبیه رنگ ارائه می دهد. این کلاس ها عبارتند از: LinearGradient، RadialGradient و SweepGradient

به منظور استفاده از Shader لازم است آن را از طریق متد () setShader به آبجکت Paint انتساب دهید.

در صورتی که ناحیه‌ی پر شده بزرگ‌تر از Shaders هست، آنگاه می توانید از طریق Shader tile model مشخص کنید که باقی ناحیه‌ی مربوطه چگونه پر شود. به طور مثال، ثابت edge (corner) بایستی برای پر کردن فضای اضافی مورد استفاده قرار گیرد. در حالی که ثابت Shader.TileMode.CLAMP اعلان می کند که تصویر مورد نظر بایستی قرینه سازی شده و نیز به اندروید اعلان می کند که عکسی را تکرار کند.

## view 5-26-4- ذخیره‌ی دائمی و ماندگارسازی داده‌های

اغلب view های متعارف اندروید قادرند داده های مربوط به وضعیت خود را نگه دارند. این داده ها سپس توسط سیستم به صورت دائمی ذخیره می شوند. سیستم اندروید جهت ذخیره‌ی داده های مربوط به وضعیت view متد onSaveInstanceState() را صدا می زند و جهت ذخیره‌ی دائمی و بازگردانی این داده ها متد onRestoreInstanceState(Parcelable) را فراخوانی می کند.

برای ماندگار سازی داده ها، مرسوم است که `View.BasedSavedState` را داخل `view` به صورت یک کلاس درونی `static` پیاده سازی نمایید.

متدهای `View.BasedSavedState` یک قرارداد شخصی برای نگه داری داده ها می باشد.

اندروید یک `view` را بر اساس ID آن در فایل `layout` میزبانی میزبانی کرده و سپس آبجکت `Bundle` را (که حاوی داده های مربوط به وضعیت `view` می باشد) به `view` ارسال می کند. `View` با داده های کپسوله شده در این آبجکت وضعیت خود را بازگردانی می نماید.

شما می بایست اطلاعات UI را دقیقا در همان وضعیتی که کاربر آن را ترک کرده، نگه دارید و بعد مجددآ آن را بازگردانی نمایید. برای مثال موقعیت نوار پیمایش یا انتخاب کاربر را به حالت قبلی آن بازگردانید.



## بخش هفتم :

# استفاده از Resource Selector در اندروید - انتخاب منابع متناسب با هر دستگاه به وسیله ی گزینشگر منابع (resource Qualifier)

مبحث حاضر به شرح resource selector و کاربرد آن در اندروید پرداخته و سپس توضیح می دهد چگونه برنامه را طراحی نمایید که بر روی انواع نمایشگر با عرض و تراکم پیکسلی متفاوت قابل اجرا بوده و از زبان های مختلف پشتیبانی کند.

## 27- تنظیمات و پیکربندی های مختلف دستگاه های اندروید

### 27-1- طراحی برنامه برای انواع دستگاه های اندروید (با نمایشگر و اندازه های مختلف)

دستگاه های اندرویدی از نظر اندازه، تراکم پیکسلی نمایشگر و تنظیمات زبان مورد استفاده متفاوت هستند. برای سازگاری برنامه با دستگاه های مختلف، اندروید منابع مناسب با دستگاه میزبان را انتخاب نموده و اندازه بندی کامپوننت های رابط کاربری (UI) را خود بر حسب واحد dip (فرمت اندازه بندی نسبی) و به صورت اتوماتیک انجام می دهد.

### 27-2- (تعریف کننده و گزینشگر منابع) Resource qualifier

توسعه دهنده با بهره گیری از resource qualifier قادر خواهد برنامه را طوری طراحی کند که با دستگاه های مختلف سازگار باشد و در واقع برای هر دستگاه منابع مناسب (وضعیت نمایش، کیفیت تصویر و زبان مرتبط) را انتخاب نماید. برای ارائه ی فایل های منبع مناسب با تنظیمات هر دستگاه که توسط گزینشگر qualifier معرفی شده و قابل دستیابی می باشد، لازم است یک subfolder در پوشه ی res با ترکیب این qualifier و اسم فایل ایجاد نمایید. به عنوان مثال اسم فایل layout را layout-qualifier کنید.

**نکته:** سیستم اندروید تنظیمات و پیکربندی دستگاه جاری را ارزیابی نموده، سپس بر اساس نتیجه ای ارزیابی فایل مناسب را به صورت خودکار انتخاب می کند.

### Resource qualifier-5-27-3

#### استفاده از وضعیت نمایش orientation به عنوان یک گزینش منابع بر اساس وضعیت نمایش (resource qualifier)

فرض کنید که می خواهید برای activity خود در نمای افقی (landscape mode) از یک فایل layout به نام activity\_main.xml استفاده نمایید. در چنین شرایطی، ابتدا پوشه-land را ایجاد نموده، سپس فایل جدید را با همین نام (activity\_main.xml) داخل این پوشه جایگذاری نمایید.

#### Version qualifier-5-27-4

برای انتخاب منابع مناسب با دستگاه جاری، توسعه دهنده می تواند از ورژن کتابخانه های اندروید (API level) بهره بگیرد که مبتنی بر qualifier یا تعریف کننده  $v[minimun API level]$  می باشد. از این طریق شما می توانید بر اساس ویرایش اندروید theme و style متفاوت ارائه نمایید.

به عنوان نمونه برنامه نویس می تواند با تکیه بر این qualifier که منابع را بر اساس ویرایش اندروید گزینش می کند، style مربوطه را برای برنامه ارائه نماید و تجربه ی کاربری بهینه را به ارمغان بیاورد.

#### Width&Height-5-27-5

یکی دیگر از آیتم هایی که توسعه دهنگان برای گزینش منابع مناسب با هر دستگاه مورد استفاده قرار می دهند، available width و smallest available width می باشد. available width عبارت است از حداقل طول قابل استفاده ی صفحه برای نمایش کامپوننت های UI و available width عبارت است از پهنا واقعی بر اساس وضعیت نمایش دستگاه (نمای افقی/عمودی).

می توان با استفاده از گزینشگر `width`, برای مثال، بر اساس پهنهای موجود نمایشگر دستگاه، فایل `layout` مناسب را انتخاب نمود. این نوع انتخاب منبع مبتنی بر `qualifier` (گزینشگر یا تعریف کننده) (`w[Number]dp`-`sw[Number]dp`) مخفف تعداد پیکسل ها مستقل از دستگاه (`dip`) می باشد. به عنوان مثال، نمایشگر یک تبلت 7 اینچی معمولاً `600dp` می باشد و شما می توانید از طریق گزینشگر `layout-sw600dp` فایل `res/layout` سازگار با نمایشگر دستگاه میزبان را انتخاب نمایید.

## سایر `qualifier` ها جهت گزینش منابع

سایر `resource qualifier` ها برای گزینش منابع را در آدرس زیر پیدا خواهید کرد.

<http://developer.android.com/guide/topics/resources/providing-resources.html#AlternativeResources>

## String-28-5 محتواهای متñی (ترجمه های نوشته های برنامه)

می توانید با استفاده از `resource selector` ها، پوشه `i` `values` را که می تواند حامل مقادیر رشتة ای باشند، انتخاب نمایید. برای این منظور می توانید از پوشه `i` `values-qualifier` استفاده کنید.

به عنوان مثال، جهت ارائه محتواهای متñی به زبان انگلیسی یا آلمانی، می بایست از پوشه `i` `values-de` و `values-en` استفاده نمایید.

## Plural-5-28-1

اندروید از فایل های محتوایی به نام Plurals پشتیبانی می کند. Plurals ها منابعی در قالب XML هستند که به شما این امکان را می دهد تا محتوای متنی مورد نیاز را بر اساس کمیت (مقدار خصیصه) انتخاب نمایید. برای استفاده از فایل مقادیر مختلف همچون "one", "zero", "one" (quantity) انتخاب نمایید. برای استفاده از فایل XML به انتساب می دهید، سپس در کد با استفاده از متدهای getQuantityString() مقدار صحیح را از XML واکشی می کنید. می توانید رشته ها را فرمت دهی کنید. اما در صورت فرمت دهی String ها، برای بازیابی مقدار صحیح، لازم است فایل plural و number را به عنوان پارامتر به متدهای name برده ارسال کنید. چنانچه از Object ها برای فرمت دهی استفاده می کنید، در آن صورت کافی است آن ها را به عنوان پارامتر اضافی به متدهای getQuantityString() ارسال نمایید.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <plurals
        name="tutorials">
        <item quantity="zero">no Tutorial </item>
        <item quantity="one">one Tutorial </item>
        <item quantity="other">%d Tutorials</item>
    </plurals>
</resources>
```

تکه کد زیر یک plural تعریف می کند. این فایل بایستی در پوشه res/values "جاگذاری شده و در این مثال نام آن را "plurals" می باشد.

```
// number is defined somewhere before this
// number =...
// get the Resources
Resources res = getResources();
// get the
String quantityString = res.getQuantityString(R.plurals.tutorials,
    number, number);
// do something with it..
```

لازم به ذکر است که انتخاب بر اساس ضرورت نحوی و گرامری صورت گرفته است. رشته `zero` در انگلیسی کاملاً نادیده گرفته می شود حتی اگر quantity برابر 0 باشد چرا که 0 به استثنای عدد 1

هیچ تفاوتی از نظر گرامری با 2 یا سایر اعداد ندارد ("two", "one tutorial", "no tutorial") و غیره ... ("tutorials")

## 28-5-استفاده از google translate

می توانید String (محتوای متنی) خود را به صورت دستی ترجمه کنید یا از ترجمه‌ی ماشینی استفاده نمایید.

اگرچه ترجمه‌ی ماشینی چندان مناسب نیست اما برخی مواقع می‌تواند تنها گزینه پیش رو باشد به خصوص اگر بودجه‌ی اپلیکیشن مورد نظر محدود باشد. نرم افزار Google Translate به صورت رایگان ترجمه را برای شما انجام می‌دهد که طبیعتاً به خاطر ماشینی بودن آن از کیفیت پایین برخوردار است، اما در صورت تمایل می‌توانید در ازای پرداخت مبلغی، ترجمه‌ی ماشینی را توسط کارشناس ویرایش شده تحویل بگیرید.

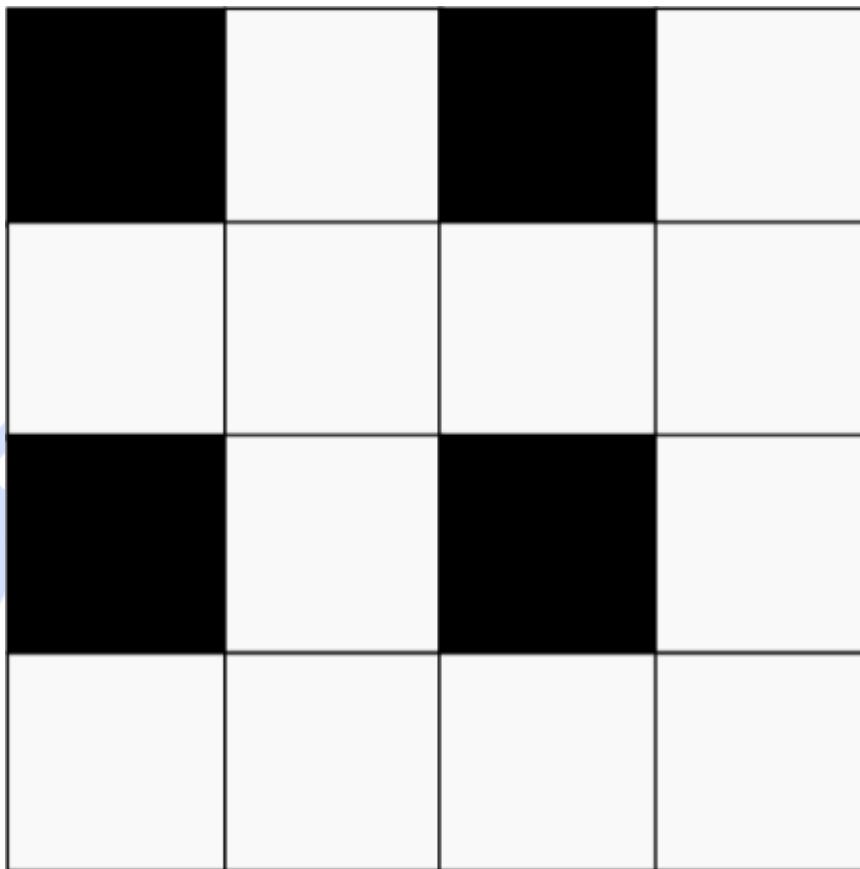
**نکته:** نرم افزار نام برده در ابتدا پیشنهاد ترجمه‌ی با کیفیت و پولی را به شما می‌دهد. پس از رد کردن این پیشنهاد می‌توانید از سرویس ترجمه‌ی ماشینی بهره بگیرید.

## 28-3-مدیریت مبحث کیفیت تصویر و تراکم پیکسلی نمایشگر های مختلف

نمایشگر دستگاه‌های اندرویدی از نظر کیفیت تصویر (resolution) و تراکم پیکسلی متفاوت هستند.

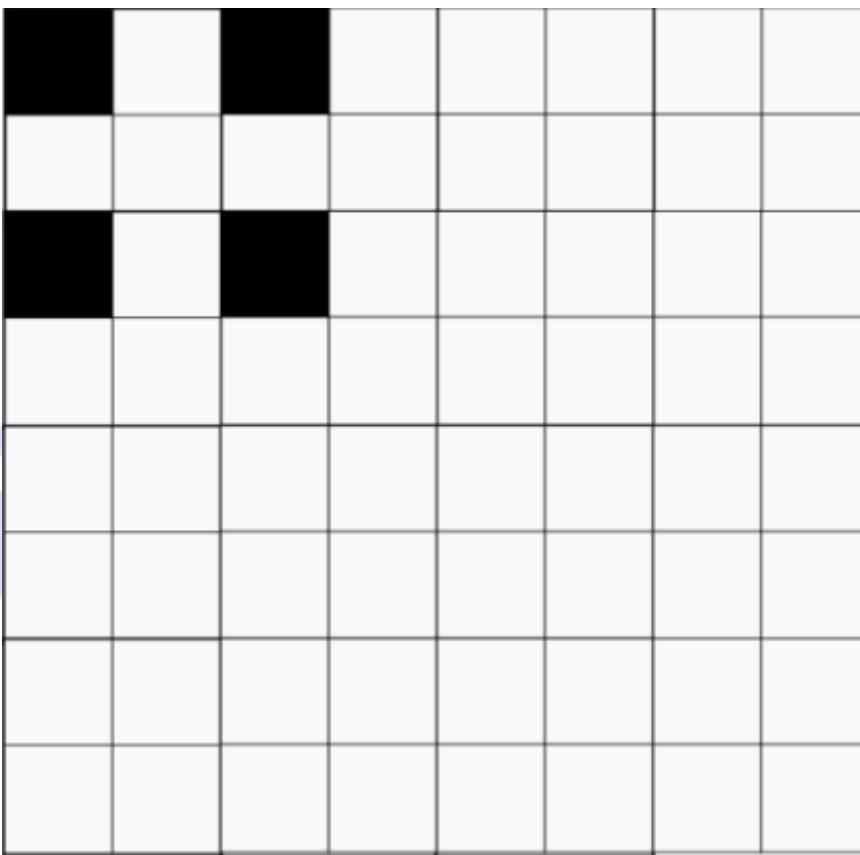
یک کامپوننت UI با اندازه‌ی یکسان می‌تواند از تراکم پیکسلی متفاوتی برخوردار بوده و به عبارت دیگر تعداد پیکسل‌های تشکیل دهنده‌ی آن بیشتر باشد که نتیجتاً کیفیت بهتر را به دنبال دارد.

به عنوان مثال، اگر تعداد پیکسل‌های یک کامپوننت رابط کاربری را خود به طور مستقیم تعیین نمایید، ظاهر آن در دستگاهی با تراکم پیکسلی پایین به صورت زیر خواهد بود.



حال چنانچه تصویر فوق را با همین تعداد پیکسل در دستگاهی با تراکم پیکسلی بالاتر مشاهده کنید، خواهید دید که ظاهر و رابط کاربری اپلیکیشن ناخوانا و غیر قابل استفاده خواهد بود.

آموزشگاه حکیمکرداده



به همین علت، شما بایستی منابع و محتوای گرافیکی لازم و مناسب (همچون آیکون اجرای برنامه/launcher و action bar) را در وضوح مختلف ارائه نمایید. این کار را می‌توانید با استفاده از pixel density به عنوان گزینش گر منابع مورد نیاز به انجام برسانید.

#### 28-4- استفاده از density به عنوان گزینش گر منابع (انتخاب محتوا بر اساس چگالی پیکسلی)

می‌توانید pixel density را مبنای گزینش منابع مورد نیاز اپلیکیشن خود قرار دهید. اندروید برای این منظور گزینه‌های زیر را در اختیار توسعه دهنده قرار می‌دهد. حداقل (baseline) Drawable 160dpi است. چنانچه دستگاه اندروید مورد نظر 320dpi داشته باشد، در آن صورت از drawable-hdpi انتخاب می‌شود.

**Table 1. Density resource selector**

گزینشگر بر اساس پیکسل	معادل
ldpi	160 dpi x 0.75
mdpi	160 dpi
hdpi	1.5 x 160 dpi = 240 dpi
xhdpi	2 x 160 dpi = 320 dpi
xxhdpi	3 x 160 dpi = 480 dpi
xxxhdpi	

## ۵-۲۸-۵- ارائه‌ی آیکون در اندازه‌های مختلف

شما بایستی launcher یا آیکون اجرا، آیکون مربوط به action bar و آیکونی که در نوار اعلانات (notification bar) به نمایش در می‌آید را در پنج اندازه‌ی مهم زیر ارائه نمایید.

**Table 2. Android icons size**

Icons	mdpi	hdpi	xhdpi	xxhdpi	xxxhdpi
Launcher icon	48 px	72 px	96 px	144 px	192 px
Action bar icon	32 px	48 px	64 px	96 px	128 px
Notification icon	24 px	36 px	48 px	72 px	96 px

لازم به ذکر است که آیکون اجرای اپلیکیشن (launcher) را بایستی برای ارائه در Google Play در اندازه‌ی 512x512 px نیز در نظر بگیرید.

## 29-5-تعیین اندازه های کامپوننت های ال در فایل های layout

### 29-5-اندازه بندی در ابعاد ثابت (fixed) یا نسبی (relative)

اندروید به شما این امکان را می دهد تا اندازه های کامپوننت های رابط کاربری را در ابعاد ثابت یا نسبی داخل فایل های layout مشخص نمایید. در صورت تعیین اندازه های المان های رابط کاربری در اندازه هی ثابت، برای مثال پیکسل، ظاهر اپلیکیشن شما ممکن است در یک دستگاه مناسب باشد اما در دستگاهی که دارای تراکم پیکسلی بالاتر است، دکمه به نسبت نمایشگر میزبان بسیار کوچک و بی تناسب جلوه کند.

در ساخت اپلیکیشن های اندرویدی توصیه می شود همیشه ابعاد کامپوننت های ال خود را به صورت نسبی (relative) مشخص نمایید.

### 29-5-استفاده از واحد dp برای اندازه بندی و تعیین ابعاد المان های ال به صورت نسبی

واحد اندازه بندی که بایستی در تعیین ابعاد المان های رابط کاربری مورد استفاده قرار دهید، dp می باشد.

**توجه:** dp حالت اختصار dip یا device independent pixel به معنی تعیین تعداد پیکسل مستقل از چگالی دستگاه میزبان می باشد.

منظور از dp یک پیکسل در دستگاه اندروید با چگالی 160dpi می باشد. تراکم پیکسلی مزبور در واقع به چگالی اولین دستگاه تولیده شده هی اندروید (G1) اشاره دارد که از آن تحت عنوان mdpi (medium dots per inch) نیز یاد می شود.

اگر اندازه هی دستگاه را بر حسب dp مشخص نمایید، در آن صورت سیستم اندروید المان های رابط کاربری شما را به صورت خودکار مناسب با دستگاه میزبان بزرگ/کوچک و مقیاس دهی می کند.

در یک دستگاه mdpi (دستگاهی با چگالی یا تراکم پیکسلی متوسط)، 1dp معادل یک پیکسل است. این در حالی است که 1dp در دستگاهی با ldpi (چگالی پایین) کوچکتر (حدودا 120dpi)، در دستگاهی با چگالی بالا (high density) بزرگتر (حدودا 240dpi) می باشد. از این رو یک dp حدودا همان میزان فضای را در تمامی دستگاه ها اشغال می کند.

می توانید اندازه و ابعاد المان های رابط کاربری خود را در فایل های layout بر حسب dp تعیین نمایید.

### 3-29-5- اندازه بندی نوشته ها بر حسب واحد sp

چنانچه قرار است واحد اندازه بندی با تنظیمات انتخابی کاربر در خصوص نوشته ها مقیاس دهی و تنظیم شود، لازم است از واحد اندازه گیری sp استفاده نمایید. این واحد اندازه گیری شبیه به dp عمل می کند، با این تفاوت که بر اساس تنظیمات دلخواه کاربر نیز قابل مقیاس بندی شده و با آن هماهنگ می باشد (به عبارت دیگر با تنظیمات اندازه ی فونتی که کاربر تعیین می کند منطبق می باشد. از این رو خروجی با چگالی نمایشگر میزبان و انتخاب کاربر همگام خواهد بود).

چنانچه کاربر تصمیم بگیرد که اندازه ی فونت را در تنظیمات افزایش دهد، به دنبال آن تمامی view هایی که بر حسب sp تنظیم شده اند، به همان ترتیب مقیاس دهی می شوند.

### 5-30- تعیین اندازه ی کامپوننت های UI در کد برنامه (code)

API و مجموعه توابع اندروید اغلب از توسعه دهنده درخواست می کند که اندازه‌ی المان‌ها را بر حسب پیکسل مشخص کند و واحد dp را به عنوان ورودی از شما نمی‌پذیرد. در چنین شرایطی لازم است واحد dp را به پیکسل تبدیل نمایید.

برای نیل به این هدف می‌توانید از متدهای زیر استفاده نمایید. این متدهای ابعاد المان را بر حسب dp به عنوان ورودی دریافت می‌کنند سپس آن را داخل بدنۀ خود به پیکسل تبدیل نموده و در خروجی برمی‌گردانند.

```
public int convertToPixelFromDp(int dpInput) {
    // get the screen's density scale
    final float scale = getResources().getDisplayMetrics().density;
    // convert the dps to pixels, based on density scale
    return (int) (dpInput * scale + 0.5f);
}
```

جهت بدست آوردن چگالی پیکسلی دستگاه جاری می‌توانید متدهای زیر را مورد استفاده قرار دهید.

```
getResources().getConfiguration().densityDpi;
```

## بخش هشتم :

# پیاده سازی انیمیشن های اندرویدی/آموزش Animation API

مبخت حاضر به شرح نحوه ی پیاده سازی انیمیشن در اندروید (استفاده از property Animation API) می پردازد. پروژه های این بخش در محیط کاری Android Studio نوشته شده و مبتنی بر ویرایش 5.1 اندروید می باشد.

## 31-5-انیمیشن سازی در اندروید

### 31-5-1-استفاده از انیمیشن در اندروید

ویرایش 3.0 اندروید ابزار Properties Animation API را برای اولین بار رونمایی کرد که به توسعه دهنده این امکان را می دهد تا مقادیر property های یک آبجکت را از یک نقطه ای زمانی تا نقطه ای زمانی دیگر ویرایش نموده و جلوه های دیداری زیبایی را خلق کند.

با بهره گیری از API مذکور قادر خواهید برای property های دلخواه آبجکت یک مقدار آغاز و یک مقدار پایان تعیین کنید. این دو مقدار یک بازه ای زمانی مشخص می کنند که طی آن ظاهر المان تغییر کرده یا از مکانی در صفحه به مکان دیگری حرکت می کند.

این API صرفاً محدود به Views نبوده و به راحتی برای سایر آبجکت های جاوا قابل استفاده می باشد.

### 31-5-2-کلاس های AnimatorListener و Animator

کلاس پدر (superclass) توابع انیمیشن سازی اندروید (Animation API) کلاس Animator می باشد. Animator کلاسی است که توسعه دهنده برای ویرایش attribute های یک آبجکت مورد استفاده قرار می دهد.

همچنین می توانید یک نمونه از کلاس AnimatorListener را به کلاس Animator اضافه نمایید.  
کلاس نام بردۀ یک گوش فرخوان/listener است و این قابلیت را دارد که در طی مراحل مختلف یک اینیمیشن فراخوانی شود.

شما می توانید با استفاده از این listener به اتفاقات معینی گوش داده و قبل یا بعد از رخداد اتفاق مورد نظر، عملیات معینی را به اجرا بگذارید. به عنوان مثال یک view را از ViewGroup حذف نمایید یا ظاهر یک المان آلا را در بازه‌ی زمانی معینی تغییر دهید.

### 31-5-ViewPropertyAnimator-کلاس

کلاس ViewPropertyAnimator در ویرایش 3.1 اندروید ارائه شده و دسترسی به اینیمیشن‌های متعارف که بر روی view‌ها اجرا می‌شوند را به مرتب آسان تر ساخت.

متدهای animate() بر روی view فراخوانی شده و نمونه‌ای از ViewPropertyAnimator را در خروجی گرداند. این آبجکت به شما امکان اجرای چندین اینیمیشن را به طور همزمان می‌دهد. آبجکت ViewPropertyAnimator از fluent API برای پشتیبانی از کدهای خوانا، نوشتمن روان و پشت سرهم دستورات استفاده کرده و علاوه بر آن به شما این اجازه را می‌دهد تا مدت زمان اجرای اینیمیشن را مشخص نمایید (Fluent API عبارت است از پیاده‌سازی interface شی‌گرا به صورتی که قابلیت خوانایی کد به طور چشم‌گیری افزایش می‌یابد و طوری که برنامه نویس بتواند دستورات را به صورت روان و پشت سرهم یکجا بنویسد).

هدف از بکار بردن ViewPropertyAnimator ارائه‌ی یک API ساده برای اجرا و دسترسی آسان به اینیمیشن‌های پرکاربرد اندروید می‌باشد.

کد زیر نحوه‌ی استفاده از این متدهای ViewPropertyAnimator را به نمایش می‌گذارد.

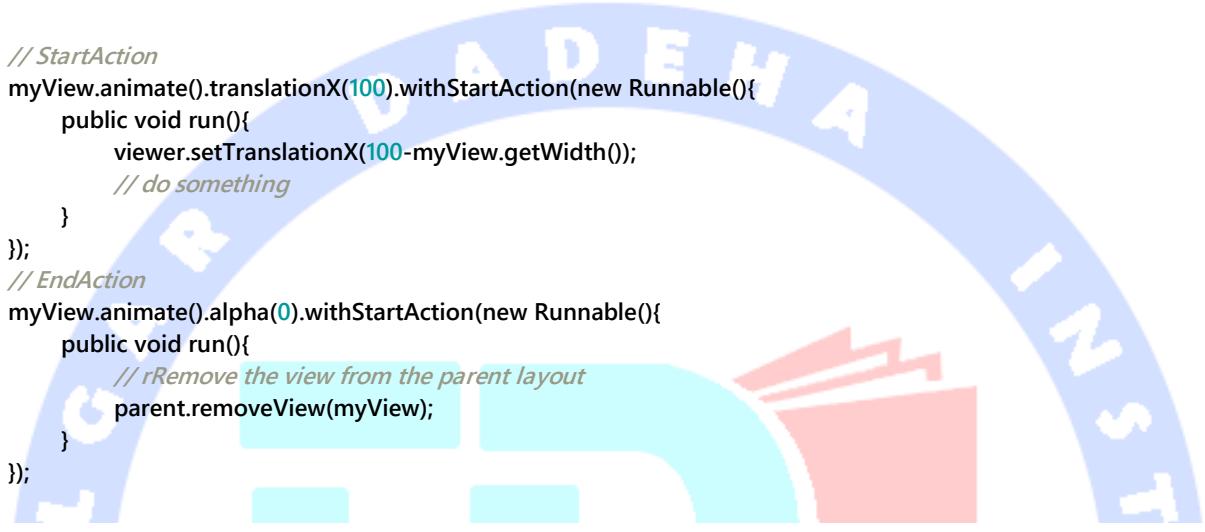
```
// Using hardware layer
myView.animate().translationX(400).withLayer();
```

به منظور افزایش کارایی، شما می‌توانید به ViewPropertyAnimator این اجازه را بدهید تا از یک لایه‌ی سخت افزاری نیز استفاده کند.

```
// Using hardware layer
myView.animate().translationX(400).withLayer();
```

در صورت نیاز می توانید به طور مسقیم اینترفیس Runnable را پیاده سازی کنید که در ابتدا و انتهای انیمیشن اجرا می شود.

```
// StartAction
myView.animate().translationX(100).withStartAction(new Runnable(){
    public void run(){
        viewer.setTranslationX(100-myView.getWidth());
        // do something
    }
});
// EndAction
myView.animate().alpha(0).withEndAction(new Runnable(){
    public void run(){
        // Remove the view from the parent layout
        parent.removeView(myView);
    }
});
```



متدهای setInterpolator() و setEvaluator() به شما اجازه می دهد تا آبجکتی از جنس TimeInterpolator را تعریف کرده و تغییر مقدار را در طول زمان مشخصی انجام دهید. حالت استاندارد linear می باشد.

محیط اندروید (platform) تعدادی حالت پیش فرض و استاندارد همچون AccelerateDecelerateInterpolator تعیین می کند که در آن انیمیشن با سرعت پایین آغاز شده، در اواسط تندرید می شود و بار دیگر در پایان با سرعت پایین خاتمه می یابد.

از طریق متدهای setEvaluator() و setInterpolator() می توانید آبجکتی از نوع TypeEvaluator را تعریف کنید که به شما این امکان را می دهد تا برای property type دلخواه انیمیشن تعریف کنید. این کار با ارائه یک

های اختصاصی برای نوع هایی انجام می شود که اندروید آن ها را نمی شناسد و به صورت پیش فرض توسط سیستم اینیمیشن سازی اندروید استفاده نمی شود.

## Layout animations-5-31-4

کلاس layout container این قابلیت را به توسعه دهنده می دهد تا بر روی LayoutTransition (ظرف و میزبان layout) اینیمیشن تنظیم کند. بدین وسیله هر تغییر در view hierarchy این view hierarchy اینیمیشن اجرا خواهد شد.

```
package com.example.android.layoutanimation;
import android.animation.LayoutTransition;
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
public class MainActivity extends Activity {
    private ViewGroup viewGroup;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        LayoutTransition l = new LayoutTransition();
        l.enableTransitionType(LayoutTransition.CHANGING);
        viewGroup = (ViewGroup) findViewById(R.id.container);
        viewGroup.setLayoutTransition(l);
    }
    public void onClick(View view) {
        viewGroup.addView(new Button(this));
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.activity_main, menu);
        return true;
    }
}
```

## 5-31-5- اعمال اینیمیشن بر روی انتقال بین activity ها (پیاده سازی انیمیشن بر روی transition بین دو activity)

انیمیشن را می توان علاوه بر روی Views، بر روی انتقال بین activity ها نیز اعمال نمود.

کلاس ActivityOptions به شما اجازه می دهد تا اینیمیشن های پیش فرض اندروید و در صورت نیاز اینیمیشن های اختصاصی خود را پیاده سازی نمایید.

```
public void onClick(View view) {
    Intent intent = new Intent(this, SecondActivity.class);
    ActivityOptions options = ActivityOptions.makeScaleUpAnimation(view, 0,
            0, view.getWidth(), view.getHeight());
    startActivity(intent, options.toBundle());
}
```

## آموزش View Animation

این بخش به شرح نحوه استفاده از Properties animation API می پردازد.

یک پروژه می باشد که در آن activity جدید اندروید به ترتیب به نام های AnimationExampleActivity و com.vogella.android.animation.views می باشد. محتوای آن را به صورت زیر ویرایش نمایید.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <LinearLayout
        android:id="@+id/test"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" >
        <Button
            android:id="@+id/Button01"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:onClick="startAnimation"
            android:text="Rotate" />
        <Button
            android:id="@+id/Button04"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:onClick="startAnimation"
            android:text="Group" />
    </Button>
    <Button
        android:id="@+id/Button03"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="startAnimation"
        android:text="Fade" />
```

```

<Button
    android:id="@+id/Button02"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="startAnimation"
    android:text="Animate" />
</LinearLayout>
<ImageView
    android:id="@+id/imageView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_centerVertical="true"
    android:src="@drawable/icon" />
<TextView
    android:id="@+id/textView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_above="@+id/imageView1"
    android:layout_alignRight="@+id/imageView1"
    android:layout_marginBottom="30dp"
    android:text="Large Text"    android:textAppearance="?android:attr/textAppearanceLarge" />
</RelativeLayout>

```

فایل XML زیر را برای تعریف منو ایجاد نمایید.

```

<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
    <item
        android:id="@+id/item1"
        android:showAsAction="ifRoom"
        android:title="Game">
    </item>
</menu>

```

بدنے ی activity خود را به صورت زیر ویرایش نمایید.

```

package com.vogella.android.animation.views;
import android.animation.AnimatorSet;
import android.animation.ObjectAnimator;
import android.app.Activity;
import android.content.Intent;
import android.graphics.Paint;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.ImageView;

```

```

import android.widget.TextView;
public class AnimationExampleActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
    public void startAnimation(View view) {
        float dest = 0;
        ImageView aniView = (ImageView) findViewById(R.id.imageView1);
        switch (view.getId()) {
            case R.id.Button01:
                dest = 360;
                if (aniView.getRotation() == 360) {
                    System.out.println(aniView.getAlpha());
                    dest = 0;
                }
                ObjectAnimator animation1 = ObjectAnimator.ofFloat(aniView,
                        "rotation", dest);
                animation1.setDuration(2000);
                animation1.start();
                // Show how to load an animation from XML
                // Animation animation1 = AnimationUtils.loadAnimation(this,
                // R.anim.myanimation);
                // animation1.setAnimationListener(this);
                // animatedView1.startAnimation(animation1);
                break;
            case R.id.Button02:
                // shows how to define a animation via code
                // also use an Interpolator (BounceInterpolator)
                Paint paint = new Paint();
                TextView aniTextView = (TextView) findViewById(R.id.textView1);
                float measureTextCenter = paint.measureText(aniTextView.getText()
                        .toString());
                dest = 0 - measureTextCenter;
                if (aniTextView.getX() < 0) {
                    dest = 0;
                }
                ObjectAnimator animation2 = ObjectAnimator.ofFloat(aniTextView,
                        "x", dest);
                animation2.setDuration(2000);
                animation2.start();
                break;
            case R.id.Button03:
                // demonstrate fading and adding an AnimationListener
                dest = 1;
                if (aniView.getAlpha() > 0) {
                    dest = 0;
                }
                ObjectAnimator animation3 = ObjectAnimator.ofFloat(aniView,

```

```

        "alpha", dest);
    animation3.setDuration(2000);
    animation3.start();
    break;
case R.id.Button04:
    ObjectAnimator fadeOut = ObjectAnimator.ofFloat(aniView, "alpha",
        0f);
    fadeOut.setDuration(2000);
    ObjectAnimator mover = ObjectAnimator.ofFloat(aniView,
        "translationX", -500f, 0f);
    mover.setDuration(2000);
    ObjectAnimator fadeIn = ObjectAnimator.ofFloat(aniView, "alpha",
        0f, 1f);
    fadeIn.setDuration(2000);
    AnimatorSet animatorSet = new AnimatorSet();
    animatorSet.play(mover).with(fadeIn).after(fadeOut);
    animatorSet.start();
    break;
default:
    break;
}
}
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.mymenu, menu);
    return super.onCreateOptionsMenu(menu);
}
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    Intent intent = new Intent(this, HitActivity.class);
    startActivity(intent);
    return true;
}
}

```

یک دیگر به نام HitActivity ایجاد نمایید.

```

package com.vogella.android.animation.views;
import java.util.Random;
import android.animation.Animator;
import android.animation.AnimatorListenerAdapter;
import android.animation.AnimatorSet;
import android.animation.ObjectAnimator;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
public class HitActivity extends Activity {
    private ObjectAnimator animation1;
    private ObjectAnimator animation2;
    private Button button;

```

```

private Random random;
private int width;
private int height;
private AnimatorSet set;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.target);
    width = getWindowManager().getDefaultDisplay().getWidth();
    height = getWindowManager().getDefaultDisplay().getHeight();
    random = new Random();
    set = createAnimation();
    set.start();
    set.addListener(new AnimatorListenerAdapter() {
        @Override
        public void onAnimationEnd(Animator animation) {
            int nextX = random.nextInt(width);
            int nextY = random.nextInt(height);
            animation1 = ObjectAnimator.ofFloat(button, "x", button.getX(),
                nextX);
            animation1.setDuration(1400);
            animation2 = ObjectAnimator.ofFloat(button, "y", button.getY(),
                nextY);
            animation2.setDuration(1400);
            set.playTogether(animation1, animation2);
            set.start();
        }
    });
}
public void onClick(View view) {
    String string = button.getText().toString();
    int hitTarget = Integer.valueOf(string) + 1;
    button.setText(String.valueOf(hitTarget));
}
private AnimatorSet createAnimation() {
    int nextX = random.nextInt(width);
    int nextY = random.nextInt(height);
    button = (Button) findViewById(R.id.button1);
    animation1 = ObjectAnimator.ofFloat(button, "x", nextX);
    animation1.setDuration(1400);
    animation2 = ObjectAnimator.ofFloat(button, "y", nextY);
    animation2.setDuration(1400);
    AnimatorSet set = new AnimatorSet();
    set.playTogether(animation1, animation2);
    return set;
}
}

```

زمانی که این مثال را اجرا کرده و بر روی دکمه های مختلف آن کلیک می کنید، انیمیشن آغاز می شود. از طریق ActionBar می توانید به دوم پیمایش کنید.

## 32-5- اعمال انیمیشن بر روی انتقال بین activity ها در اندروید با shared view ها

ویرایش 5.0 اندروید این امکان را فراهم آورد تا بر روی فعل انتقال بین دو activity انیمیشن اعمال نموده و نیز view های تعریف کرد که بین دو activity مشترک هستند. پس از تعریف یک بخش مشترک، view قدیمی طی یک انیمیشن از اول در جایگاه و اندازه‌ی view جدید در activity مشترک، دوم قرار می‌گیرد.

به منظور تست این قابلیت، یک پروژه‌ی جدید ایجاد نموده و پکیج به عنوان com.vogella.android.activityanimationwithsharedviews درج نمایید. (top level package)

دو ImageView با دو فایل layout تعریف نمایید که هر دو در بردارنده‌ی یک المان android:transitionName (property) داشته باشند.

### activity\_main.xml

```
<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:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">
    <ImageView
        android:id="@+id/sharedimage"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:scaleType="centerCrop"
```

```

    android:src="@drawable/ic_sharedimage"
/>
</LinearLayout>

```

### activity\_second.xml

```

<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"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.vogella.android.activityanimationwithsharedviews.SecondActivity">
    <ImageView
        android:id="@+id/sharedimage"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:src="@drawable/ic_sharedimage"
        android:layout_alignParentBottom="true"
        android:layout_alignParentEnd="true" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world"
        android:id="@+id/textView" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="New Button"
        android:id="@+id/button"
        android:transitionName="sharedImage"
        android:layout_below="@+id/textView"
        android:layout_alignParentStart="true"
        android:layout_marginTop="54dp" />
</RelativeLayout>

```

کد activity خود را به صورت زیر ویرایش نمایید.

```

package com.vogella.android.activityanimationwithsharedviews;
import android.app.Activity;
import android.app.ActivityOptions;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.ImageView;
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {

```

```
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
final ImageView sharedImage = (ImageView) findViewById(R.id.sharedimage);
sharedImage.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        //This is where the magic happens.
        // makeSceneTransitionAnimation takes a context, view,
        // a name for the target view.
        ActivityOptions options =
            ActivityOptions.
                makeSceneTransitionAnimation(MainActivity.this, sharedImage, "sharedImage");
        Intent intent = new Intent(MainActivity.this, SecondActivity.class);
        startActivityForResult(intent, options.toBundle());
    }
});
}
}
package com.vogella.android.activityanimationwithsharedviews;
import android.app.Activity;
import android.os.Bundle;
public class SecondActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_second);
    }
}
```

پس از اجرای اپلیکیشن و کلیک بر روی المان `image view`, می بینید که المان مجبور با اینیمیشن به ای که خاصیت `android:transitionName` را دارد (کنترل دکمه) منتقل می شود.



555

آدرس آموزشگاه : تهران - خیابان شریعتی - بالاتر از خیابان ملک - جنب بانک صادرات - پلاک 651 طبقه دوم - واحد 7

88146323 - 88446780 - 88146330

## فصل ششم

# Broadcast سرویس Notification و Receiver در اندروید Manager

آموزشگاه تخلیکر داده

556

آدرس آموزشگاه : تهران - خیابان شریعتی - بالاتر از خیابان ملک - جنب بانک صادرات - پلاک 651 طبقه دوم - واحد 7

88146323 - 88446780 - 88146330

## بخش اول :

# سرویس های اندروید/Service در اندروید

مبحث جاری نحوه‌ی استفاده از سرویس‌های پیش‌فرض اندروید، پیاده‌سازی و استفاده از سرویس‌های اختصاصی را شرح می‌دهد. این آموزش مبتنی بر ویرایش 6.0 اندروید می‌باشد.

## 1-6-سرویس های اندروید

### 1-6-1-سرویس چیست

یک کامپوننت نرم افزاری است که در پس زمینه اجرا شده و مستقیماً با کاربر تعامل ندارد. از آنجایی که سرویس فاقد است، طبیعتاً به چرخه‌ی حیات یک activity نیز متصل نمی‌باشد.

سرویس‌ها اغلب برای انجام عملیات تکراری و طولانی مورد استفاده قرار می‌گیرند. از جمله‌ی این عملیات می‌توان به دانلود از اینترنت، بررسی و جستجو برای داده‌های جدید، پردازش اطلاعات، بروز آوری content provider‌ها و موارد مشابه اشاره کرد.

سرویس‌ها دارای اولویت سطح بالاتری نسبت به activity‌های غیرفعال/غیرقابل مشاهده در UI هستند و از این رو احتمال اینکه اندروید آن‌ها را به صورت خودکار خاتمه دهد بسیار پایین است.

اندروید به شما این امکان را می‌دهد تا سرویس‌ها را طوری تنظیم کنید که اگر به هر دلیلی مجبور به حذف این سرویس‌ها از حافظه شد، به مجرد قرار گرفتن منابع کافی در اختیار سیستم، قادر باشد آن‌ها را مجدداً راه اندازی کند.

می‌توان به سرویس‌ها اولویت یکسان و برابر با ACTIVITY‌های حاضر در پیش‌زمینه (BACKGROUND) اختصاص داد. در این سناریو لازم است یک notification قابل مشاهده و فعال در UI برای سرویس‌های مربوطه لحاظ نمایید. این روش بیشتر برای سرویس‌هایی بکار می‌رود که یک فایل ویدیویی یا موسیقی را پخش می‌کند.

## 6-1-2- سرویس ها و پردازش پس زمینه ای (background processing)

به صورت پیش فرض، سرویس در همان فرایندی اجرا می شود که thread اصلی اپلیکیشن در آن حال اجرا است. به همین جهت توسعه دهنده بایستی از پردازش ناهمزمان در سرویس استفاده نموده و task هایی که هزینه بر و سنگین هستند را در پس زمینه راه اندازی کند. یکی از الگوهایی که مکررا برای پیاده سازی سرویس بکار می رود، اجرای یک Thread جدید در سرویس جهت انجام پردازش در پس زمینه و خاتمه دادن سرویس به هنگام اتمام پردازش می باشد.

سرویس هایی که در بستر فرایند خود اپلیکیشن اجرا می شوند معمولاً تحت عنوان service های محلی یا local شناخته می شوند.

## 6-1-3- سرویس های خود محیط اندروید (platform) و سرویس های اختصاصی

محیط اندروید سرویس های آماده و از پیش تعریف شده ای را درنظر گرفته و راه اندازی می کند که تمامی اپلیکیشن های اندرویدی، در صورت برخوردار بودن از مجوزهای لازم قادر به استفاده از آن ها استفاده می باشند. سرویس های سیستم را کلاسی به نام Manager در اختیار اپلیکیشن ها قرار می دهد. کافی است برای دسترسی به آن متد getSystemService() را فراخوانی نمایید.

کلاس Context تعدادی ثابت فراهم می کند که شما با استفاده از آن ها می توانید سرویس های نام برد را فراخوانی کنید.

اپلیکیشن اندروید می تواند علاوه بر سرویس های پیش فرض سیستم اندروید، سرویس های اختصاصی تعریف نموده و از آن ها در کنار سرویس های سیستم استفاده نماید.

توسعه دهنده قادر است با پیاده سازی سرویس های اختصاصی خود اپلیکیشن های پاسخگو و تعاملی (responsive) طراحی نماید. شما می توانید داده های اپلیکیشن را به وسیله های سرویس واکنش نموده و زمانی که اپلیکیشن راه اندازی شد، داده های جدید در اختیار کاربر قرار دهید.

## DADEH

### 6-1-4-راه اندازی و تعریف سرویس های اختصاصی

سرویس های اختصاصی اغلب توسط کامپوننت های دیگر راه اندازی می شوند، به عبارت دیگر سایر اجزا نرم افزاری اپلیکیشن های اندرویدی نظیر activity ها، broadcast receiver ها و سرویس های دیگر هستند که سرویس های اختصاصی را راه اندازی می کنند.

### 6-1-5-سرویس های پیش زمینه (foreground)

سرویس پیش زمینه سرویسی است که از نظر اولویت و اهمیت با یک activity فعال و قابل مشاهده در UI یکسان است و به همین جهت حتی اگر سیستم اندروید با کمبود حافظه مواجه باشد باز هم اجازه‌ی حذف از آن ها حافظه را ندارد. سرویس foreground می بایست در نوار نشان دهنده‌ی وضعیت کلی سیستم (status bar) یک اطلاعیه یا notification در زیر بخش عنوان "Ongoing" مختص به خود داشته باشد. این بدین معنی است که تا زمان حذف سرویس از یا حافظه، notification قابل dismiss و حذف از status bar نخواهد بود.

```
Notification notification = new Notification(R.drawable.icon, getText(R.string.ticker_text),
    System.currentTimeMillis());
Intent notificationIntent = new Intent(this, ExampleActivity.class);
PendingIntent pendingIntent = PendingIntent.getActivity(this, 0, notificationIntent, 0);
notification.setLatestEventInfo(this, getText(R.string.notification_title),
    getText(R.string.notification_message), pendingIntent);
startForeground(ONGOING_NOTIFICATION_ID, notification);
```

### 6-2-تعریف سرویس های اختصاصی

## 2-6-پیاده سازی و اعلان

برای پیاده سازی یک سرویس اختصاصی، ابتدا لازم است آن را با استفاده از تگ service در لایه XML (فایل تنظیمات AndroidManifest.xml) تعریف نموده و سپس در کدهای جاوا (کلاسی که سرویس را پیاده سازی می کند) از کلاس های مشتق آن ارث بری نمایید.

کد زیر نحوه تعریف سرویس در لایه XML و پیاده سازی آن در کلاس های جاوا را نمایش می دهد.

```
<service
    android:name="MyService"
    android:icon="@drawable/icon"
    android:label="@string/service_name"
    >
</service>
public class MyService extends Service {
    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        //TODO do something useful
        return Service.START_NOT_STICKY;
    }
    @Override
    public IBinder onBind(Intent intent) {
        //TODO for communication return IBinder implementation
        return null;
    }
}
```

**آراه اندازی یک سرویس**

دیگر کامپوننت های نرم افزاری اندروید (activity, receiver, service) می توانند با فراخوانی تابع startService(intent)، یک سرویس اندروید را راه اندازی کنند.

```
// use this to start and trigger a service
Intent i= new Intent(context, MyService.class);
// potentially add data to the intent
i.putExtra("KEY1", "Value to be used by the service");
context.startService(i);
```

روش دیگر راه اندازی سرویس، فراخوانی متده است bindService() می باشد. از این طریق شما می توانید مستقیماً با service مورد نظر تعامل داشته باشید.

در آینده بیشتر درباره ای روشن دوم توضیح خواهیم داد.

## 2-2-6-فرایند راه اندازی و اجرای سرویس

چنانچه پس از فراخوانی متده startService(intent)، سرویس همچنان راه اندازی نشده باشد، آنگاه آبجکت service ایجاد شده و متده onCreate() از کلاس سرویس فراخوانی می شود.

پس از اجرای سرویس، متده onStartCommand(intent) در سرویس فراخوانی می شود. سپس آبجکت Intent را به عنوان پارامتر ازتابع فراخوانده شده ای startService(intent) دریافت می کند.

در واقع سیستم اندروید با فراخوانی متده startService()، تابع onStartCommand(intent) را زمانی صدا می زند که کامپوننت نرم افزاری دیگری همچون activity نیاز به اجرای سرویس مورد نظر داشته باشد. لازم به ذکر است که اگر برنامه نویس این متده را پیاده سازی کند، آنگاه مسئولیت متوقف نمودن سرویس را بایستی خود با صدا زدن توابع stopSelf()، stopService() بر عهده بگیرد.

اگر (startService(intent) صدا خورده شود، درحالی که سرویس همچنان در حال اجر است، در آن صورت متده onStartCommand() نیز فراخوانی می شود. از این جهت سرویس بایستی این قابلیت و آمادگی را داشته باشد که متده onStartCommand() بارها صدا خورده شود.

**نکته:** در صورتی که این متده دوبار در کد فراخوانی شود، چه اتفاق رخ خواهد داد؟ آیا لازم است نگران همگام سازی فرخوانی های () onStartCommand() باشید؟ در پاسخ باید گفت خیر. این متده را سیستم اندروید در thread اصلی (UI) صدا می زند، از این جهت امکان فراخوانی همزمان آن در دو thread متفاوت وجود ندارد.

سرویس تنها یکبار راه اندازی می شود، فارغ تعداد دفعاتی که متد `startService()` فراخوانی می شود.

### 3-2-6-رفتار بازآغازی و restart سرویس

سرویس داخل بدنہ ی متد `onStartCommand()` خود یک مقدار `int` را بر می گرداند `(return)`. این دستور رفتار سرویس را در صورتی که محیط اندروید `(platform)` مجبور به خاتمه دادن سرویس و حذف آن از حافظه شود را مشخص می نماید. برای این منظور می توانید از ثوابت زیر استفاده نمایید. این ثوابت به همراه شرح کاربرد در جدول زیر ذکر شده اند.

Table 1. Restart options

ثابت	شرح
<code>Service.START_STICKY</code>	اگر سیستم سرویس را از حافظه حذف کرد، سرویس مجدد راه اندازی می شود. آبجکت <code>Intent</code> ارسالی به متد <code>onStartCommand</code> حامل مقدار <code>null</code> خواهد بود. این گزینه اغلب برای سرویس هایی بکار می روند که خود اطلاعات مربوط به وضعیتشان را مدیریت کرده و وابستگی به داده های موجود در <code>Intent</code> ندارند.
<code>Service.START_NOT_STICKY</code>	سرویس راه اندازی نشده است. برای سرویس هایی مورد استفاده

Table 1. Restart options

ثابت	شرح
	قرار می گیرد در فواصل زمانی معین راه اندازی می شوند. سرویس تنها هنگامی راه اندازی می شود که runtime تعدادی فراخوانی () startService() را از زمان حذف سرویس از حافظه، به صورت معلق و آماده ای اجرا داشته باشد.
Service.START_REDELIVER_INTENT	عملکردی مشابه Service.START_STICKY دارد با این تفاوت که آبجکت intent اصلی onStartCommand مجدداً به متند ارسال می شود.

**نکته:** می توانید با فراخوانی متند Intent.getFlags() به همراه هر یک از پارامترهای 1. با سرویس = START\_FLAG\_REDELIVERY 2. Service.START\_REDELIVER\_INTENT راه اندازی شده باشد 3. Service.START\_REDELIVER\_INTENT صدا خورده شده باشد. اگر سرویس با START\_FLAG\_RETRY فراخوانده شده باشد (بررسی کنید آیا سرویس مجدد راه اندازی شده است یا خیر).

## ۶-۲-۴- متوقف کردن یک سرویس

برای متوقف کردن یک سرویس کافی است متند stopService() را صدا بزنید. فارغ از تعداد دفعاتی که متند startService() را برای راه اندازی خود سرویس فراخوانی کردید، یکبار صدا زدن تابع stopService() سرویس مورد نظر را کاملاً متوقف می سازد.

یک سرویس می تواند خود را با فراخوانی متدها stopSelf() متوقف کند. این تابع معمولاً زمانی صدای خورده می شود که سرویس کارش کاملاً تمام شده باشد.

استفاده از IntentServices (ارت بری از کلاس IntentServices) می توانید از کلاس IntentService برای پیاده سازی سرویس استفاده نمایید.

برای اجرای تسلیم های خاصی در background بکار می رود. نمونه ای پیاده سازی شده از این کلاس پس از اتمام وظایفش، خود را به صورت اتوماتیک متوقف ساخته و از حافظه حذف می نماید. یکی از موارد استفاده ای این کلاس در بارگیری منابع و محتواهای خاص از اینترنت است.

کلاس IntentService متدها onHandleIntent() را ارائه می دهد. این متده را سیستم اندروید به صورت ناهمزمان (asynch) فراخوانی می نماید.

### 3-3-6- متصل کردن دو طرفه ای سرویس ها (service binding)

#### 3-3-6-1- وصل شدن از activity ها به سرویس ها

چنانچه activity لازم دارد به طور مستقیم با service تبادل داده و تعامل داشته باشد، ابتدا متده bindService() را برای راه اندازی service می زند. این متده یک آبجکت ServiceConnection را برای راه اندازی service می زند. آنگاه آغاز سرویس و اتمام اجرای متده onBind() فراخوانی می شود. متده ذکر شده یک آبجکت IBinder را به ServiceConnection برمی گرداند.

در واقع سیستم با فراخوانی onBind() متده bindService() را زمانی صدا می زند که کامپوننت دیگری درخواست اتصال به این سرویس را داشته باشد. در صورتی که توسعه دهنده این متده را پیاده سازی کرده باشد، آنگاه می بایست یک interface نیز پیاده سازی کرده و آبجکت IBinder را به عنوان خروجی برگرداند که کلاینت برای ارتباط با سرور از آن استفاده می کند. لازم به توضیح است که پیاده سازی این متده ضروری می باشد، اما چنانچه نمی خواهید اجازه ای اتصال دو طرفه (Bind) را بدهید، می توانید null را بازگردانی کنید.

با استفاده از آبجکت `IBinder` با سرویس اطلاعات رد و بدل می کنند.

زمانی که فرایند اتصال به پایان می رسد، متد `onStartCommand()` در سرویس با آبجکت `bindService()` از آن استفاده می کند.

### 6-3-2- اتصال به سرویس های محلی

اگر سرویس در فرایندی یکسان با activity اجرا شود (سرویس و activity هر دو در یک فرایند اجرا شوند)، در آن صورت می توان سرویس را به activity برگرداند. این امر زمینه ای برای activity فراهم می آورد تا متد های سرویس را مستقیماً فراخوانی کند. این روش را به صورت کاربردی در `<exercise_bindlocalservice>` می بینید که پیاده سازی شده است.

### 6-3-3- اتصال به سرویس با استفاده از IPC

اگر سرویس در فرایند ویژه‌ی خودش اجرا شود، آنگاه بایستی با استفاده از IPC با سرویس اطلاعات تبادل نمایید.

### 6-4-1- اجرای یک سرویس در فرایند مختص به خود

می توانید سرویس خود را طوری تنظیم نمایید که در فرایند جداگانه و مختص به خود اجرا شود. برای این منظور مقدار `service` را در داخل تگ `process:process_description` برابر `"@drawable/icon"` قرار می دهید.

```
<service
    android:name="WordService"
    android:process=":my_process"
    android:icon="@drawable/icon"
```

```

    android:label="@string/service_name"
  >
</service>
```

علامت دونقطه رو به روی : `android:process` به سیستم Android اعلان می کند که Service مورد نظر مختص اپلیکیشن میزبان و تعریف کننده‌ی آن است. در صورتی که از دو نقطه استفاده نشده باشد، آنگاه Service در بستر یک فرایند سراسری اجرا شده و سایر اپلیکیشن‌های اندرویدی به آن دسترسی خواهند داشت.

اجرای یک سرویس در فرایند خودش سبب مسدود شدن اپلیکیشن نمی‌شود، حتی اگر سرویس مورد نظر عملیات طولانی مدت در `thread` اصلی خود اجرا کند. اما از طرفی، از آنجایی که سرویس‌ها در فرایند خود اجرا می‌شوند، شما بایستی برای اتصال و تبادل داده با سرویس از سایر بخش‌ها از IPC بهره بگیرید.

حتی اگر سرویس در فرایند خود اجرا شود، برای دسترسی به اینترنت، برنامه نویس ملزم به استفاده از پردازش ناهمzman خواهد بود چرا که اندروید تحت هیچ عنوانی اجازه‌ی دسترسی به اینترنت را در `thread` اصلی یک فرایند نمی‌دهد.

## 2-4-6-چه زمان می‌بایست یک سرویس را در فرایند مجزا اجرا کرد؟

اگر یک سرویس را در فرایند جداگانه و مختص به خود اجرا کنید، در آن صورت یک فضا با آدرس ویژه در حافظه به آن تخصیص یافته و garbage collector دستگاه مجازی (AVD) در این فرایند، در فرایند کلی اپلیکیشن دخالتی نخواهد داشت.

اپلیکیشن به ندرت نیاز پیدا می‌کند که سرویس را در فرایند ویژه و مجزا راه اندازی کند. اجرای سرویس‌ها در بستر فرایندهای مجزا، سبب می‌شود پیاده سازی ارتباط و تبادل داده بین سایر کامپوننت‌های نرم افزاری اندروید و سرویس به تبع دشوار شود.

**نکته:** اگر می خواهید دسترسی به یک سرویس را در اختیار اپلیکیشن دیگری قرار دهید، در آن صورت سرویس بایستی در فرایند مختص به خود اجرا شود.

## 4-6-تبدیل داده و ارتباط با سرویس ها

### 4-6-1-روش های مختلف برای برقراری ارتباط با سرویس ها

راه های مختلفی برای تبدیل داده و تعامل بین activity و سرویس وجود دارد. مطالب زیر روش های ممکن برای نیل به این هدف را نام برده و روش پیشنهادی خود را در اختیار شما قرار می دهد.

### 4-6-2-استفاده از داده های کپسوله شده در intent

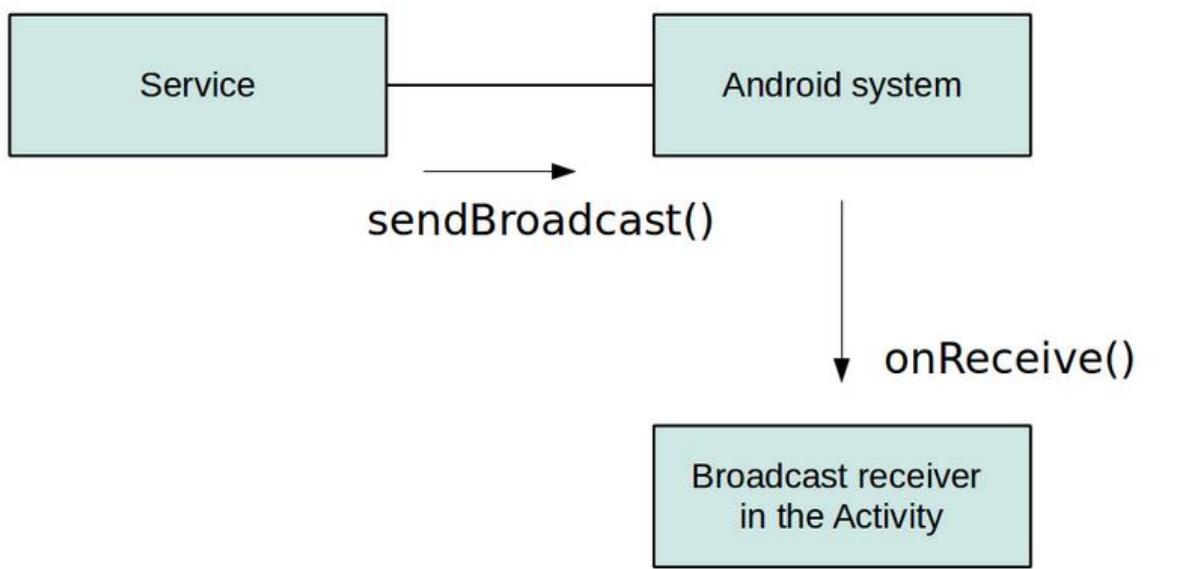
در یک سناریوی ساده، نیازی به هیچ تعامل مستقیم (بین سرویس و activity) وجود ندارد. سرویس داده های کپسوله شده در intent را از کامپوننت آغاز کننده (فراخواننده سرویس) دریافت نموده و عملیات لازم را به انجام می رساند. لازم به ذکر است که برای این منظور notification ضروری نیست. در واقع در شرایطی که سرویس محتوای یک content provider را با داده های جدید بروز آوری می کند، خود کامپوننت نرم افزاری مزبور activity را از این رخداد با خبر کرده و هیچ اقدام یا مرحله دیگری در سرویس لازم نیست. این روش هم برای سرویس های محلی و هم برای سرویس هایی که در فرایند مختص به خود اجرا می شوند، قابل پیاده سازی و استفاده خواهد بود.

### 4-6-3-receiver از

می توان برای تعامل و ارتباط بین activity و سرویس ها از receiver هایی که به این broadcast ها گوش می دهند، بهره گرفت. به عنوان مثال، activity شما می تواند یک broadcast receiver برای گوش دادن به event ای معین ثبت کند و سرویس مورد نظر اتفاق افتادن event های مربوطه را به بیرون (کاپوننت های دیگر) اعلان نماید. این روش بسیار معمول

بوده و اغلب زمانی استفاده می شود که سرویس می باشد پس از به انجام رساندن پردازش به این اتفاق را اعلان کند.

این جریان ارتباطی در تصویر زیر به تصویر کشیده شده است.



روش نام بردۀ برای سرویس های محلی و سرویس هایی که در فرایند میزبان و مختص خود اجرا می شوند، قابل استفاده می باشد.

## آموزشگاه تحلیلکرداره

### 4-4-اتصال activity به سرویس محلی

چنانچه service و activity هر دو در فرایندی یکسان اجرا شوند، در این شرایط activity قادر خواهد بود به سرویس مستقیماً به صورت دو طرفه وصل شود. روش حاضر از میان گزینه هایی که تاکنون عنوان شده، بهینه ترین بوده و برای زمانی که activity نیاز دارد با سرویس با سرعت بالا تبادل داده داشته باشد بسیار مناسب می باشد.

ناگفته نماند که این روش تنها برای سرویس های محلی (سرویس هایی که) قابل پیاده سازی و استفاده می باشد.

## Messenger با ResultReceiver و Handler\_6-4-5

چنانچه سرویس با activity تعامل دو طرفه داشته باشد (اطلاعاتی را به activity برگرداند)، در آن صورت می تواند از طریق داده های کپسوله شده در intent که از activity دریافت می کند، آبجکتی از جنس Messenger دریافت نماید. در صورتی که Messenger به Handler در activity متصل باشد، آنگاه service قادر خواهد بود آبجکت هایی از جنس Message را به activity ارسال نماید.

در واقع اینترفیس parcelable را پیاده سازی می کند، بدین معنی که می توان آن را به فرایند دیگر ارسال کرده و با استفاده از این آبجکت Message هایی را به Handler در activity ارسال نمود.

همچنین متدی به نام `getBinder()` را ارائه می دهد. این متد قابلیت ارسال آبجکتی از جنس Messenger به activity را فراهم می نماید. نیز متعاقبا قادر خواهد بود Message های (نمونه هایی از کلاس Message که حاوی توصیف و آبجکت های داده ای دلخواه می باشد) متعددی را به سرویس مورد نظر ارسال کند.

این روش برای سرویس های محلی که در فرایند خود اجرا می شوند، قابل استفاده می باشد.

## 6-4-6-اتصال به سرویس در فرایند دیگر با استفاده از AIDL

به منظور تبادل داده و اتصال (bind) به سرویسی که در فرایند دیگری در حال اجرا است، برنامه نویس بایستی از IPC (ارتباط میان پردازشی) کمک بگیرد. برای نیل به این هدف، ابتدا لازم است یک فایل AIDL ایجاد کند که تقریبا مشابه interface های جاوا می باشد با این تفاوت که پسوند آن `.aidl` بوده و تنها اجازه ای ارث بری و بسط دیگر فایل های AIDL را دارد.

توصیه می شود از این روش زمانی استفاده نمایید که لازم باشد به سرویسی که در فرایند دیگری در حال اجرا است متصل شوید. برای مثال زمانی که اپلیکیشن های دیگر درخواست استفاده از سرویس مورد نظر را داشته باشند باید از این رویکرد استفاده کرد.

## تمرین: استفاده از Service ها و تبادل داده با سرویس

در مثال زیر یک activity با کنترل دکمه پیاده سازی می کنید که از سرویس برای دانلود فایل از اینترنت استفاده می کند. زمانی که کاربر بر روی دکمه در activity کلیک می کند، دانلود اطلاعات از اینترنت آغاز می گردد. پس از به پایان رسیدن دانلود، سرویس از طریق broadcast receiver به این activity اتفاق را اعلان می کند.

در مثال کاربردی حاضر از کلاس IntentService استفاده خواهد نمود چرا که این کلاس پردازش های پس زمینه ای را به صورت خودکار مدیریت می نماید.

یک پروژه و activity جدید به ترتیب به نام های com.vogella.android.service.receiver و com.vogella.android.app.MainActivity ایجاد نمایید.

سپس کلاس زیر را برای پیاده سازی سرویس تعریف نمایید.

```
package com.vogella.android.service.receiver;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.URL;
import android.app.Activity;
import android.app.IntentService;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.os.Environment;
import android.os.Message;
import android.os.Messenger;
import android.util.Log;
public class DownloadService extends IntentService {
    private int result = Activity.RESULT_CANCELED;
```

```

public static final String URL = "urlpath";
public static final String FILENAME = "filename";
public static final String FILEPATH = "filepath";
public static final String RESULT = "result";
public static final String NOTIFICATION = "com.vogella.android.service.receiver";
public DownloadService() {
    super("DownloadService");
}
// will be called asynchronously by Android
@Override
protected void onHandleIntent(Intent intent) {
    String urlPath = intent.getStringExtra(URL);
    String fileName = intent.getStringExtra(FILENAME);
    File output = new File(Environment.getExternalStorageDirectory(),
        fileName);
    if (output.exists()) {
        output.delete();
    }
    InputStream stream = null;
    FileOutputStream fos = null;
    try {
        URL url = new URL(urlPath);
        stream = url.openConnection().getInputStream();
        InputStreamReader reader = new InputStreamReader(stream);
        fos = new FileOutputStream(output.getPath());
        int next = -1;
        while ((next = reader.read()) != -1) {
            fos.write(next);
        }
        // successfully finished
        result = Activity.RESULT_OK;
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        if (stream != null) {
            try {
                stream.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        if (fos != null) {
            try {
                fos.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
    publishResults(output.getAbsolutePath(), result);
}
private void publishResults(String outputPath, int result) {
}

```

```

Intent intent = new Intent(NOTIFICATION);
intent.putExtra(FILEPATH, outputPath);
intent.putExtra(RESULT, result);
sendBroadcast(intent);
}
}

```

این کلاس را در لایه‌ی XML، فایل تنظیمات اپلیکیشن AndroidManifest.xml اضافه نمایید. حال زمان آن رسیده تا مجوزهای لازم را برای درج داده (write) در حافظه‌ی خارجی و دسترسی به اینترنت را نیز اعطا نمایید. پس از وارد کردن تغییرات فوق، محتوای xml می‌باشد.

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.vogella.android.service.receiver"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk
        android:minSdkVersion="17"
        android:targetSdkVersion="18" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.vogella.android.service.receiver.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <service android:name="com.vogella.android.service.receiver.DownloadService" >
        </service>
    </application>
</manifest>

```

فایل layout مربوط به activity خود را به صورت زیر ویرایش نمایید.

```

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

```

```

<Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="onClick"
    android:text="Download" />
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" >
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Status: " />
    <TextView
        android:id="@+id/status"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Not started" />
</LinearLayout>
</LinearLayout>

```

کلاس MainActivity خود را به صورت زیر ویرایش نمایید.

```

package com.vogella.android.service.receiver;
import android.app.Activity;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;
public class MainActivity extends Activity {
    private TextView textView;
    private BroadcastReceiver receiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            Bundle bundle = intent.getExtras();
            if (bundle != null) {
                String string = bundle.getString(DownloadService.FILEPATH);
                int resultCode = bundle.getInt(DownloadService.RESULT);
                if (resultCode == RESULT_OK) {
                    Toast.makeText(MainActivity.this,
                            "Download complete. Download URI: " + string,
                            Toast.LENGTH_LONG).show();
                    textView.setText("Download done");
                } else {
                    Toast.makeText(MainActivity.this, "Download failed",
                            Toast.LENGTH_LONG).show();
                    textView.setText("Download failed");
                }
            }
        }
    };
}

```

```
        }
    }
};

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    textView = (TextView) findViewById(R.id.status);
}

@Override
protected void onResume() {
    super.onResume();
    registerReceiver(receiver, new IntentFilter(
        DownloadService.NOTIFICATION));
}

@Override
protected void onPause() {
    super.onPause();
    unregisterReceiver(receiver);
}

public void onClick(View view) {
    Intent intent = new Intent(this, DownloadService.class);
    // add infos for the service which file to download and where to store
    intent.putExtra(DownloadService.FILENAME, "index.html");
    intent.putExtra(DownloadService.URL,
        "http://www.vogella.com/index.html");
    startService(intent);
    textView.setText("Service started");
}
```

حال زمانی که برنامه‌ی فوق را اجرا کرده و بر روی دکمه کلیک می‌کنید، سرویس بلافارسله اقدام به بارگیری اطلاعات لازم از اینترنت می‌نماید. با اتمام دانلود، رابط کاربری بروز رسانی شده و یک پیغام Toast با اسم فایل به نمایش در می‌آید.

تنظیمات را طوری تغییر دهید که سرویس در فرایند ویژه‌ی خود اجرا شود. باز دیگر اپلیکیشن را تست کرده و اطمینان حاصل نمایید که همچنان به درستی عمل می‌کند چرا که با وارد نمودن تغییرات جدید، broadcast receiver‌ها از فرایندهای مختلف broadcast‌ها را دریافت می‌کنند.

تمرین: پیاده سازی و استفاده از سرویس محلی (local service)

تمرین حاضر نحوه ای اتصال از یک activity به سرویس محلی (سرویسی که با activity در یک فرایند اجرا می شود) را به صورت عملی نمایش می دهد.

پس از اینکه دستگاه بوت می شود، سرویس راه اندازی شده و داده هایی را در فواصل زمانی مشخص از اینترنت دانلود می نماید. حال Activity، خود را به سرویس متصل کرده تا به داده هایی که اخیرا از اینترنت بارگیری شده دسترسی داشته باشد.

یک پروژه و activity جدید به ترتیب به نام های de.vogella.android.ownservice.local و MainActivity تعریف نمایید.

کلاسی به نام LocalWordService با بدنه ای زیر ایجاد نمایید.

```
package de.vogella.android.ownservice.local;
import java.util.ArrayList;
import java.util.List;
import java.util.Random;
import android.app.Service;
import android.content.Intent;
import android.os.Binder;
import android.os.IBinder;
public class LocalWordService extends Service {
    private final IBinder mBinder = new MyBinder();
    private ArrayList<String> list = new ArrayList<String>();
    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        Random random = new Random();
        if (random.nextBoolean()) {
            list.add("Linux");
        }
        if (random.nextBoolean()) {
            list.add("Android");
        }
        if (random.nextBoolean()) {
            list.add("iPhone");
        }
        if (random.nextBoolean()) {
            list.add("Windows7");
        }
        if (list.size() >= 20) {
            list.remove(0);
        }
        return Service.START_NOT_STICKY;
    }
}
```

```

@Override
public IBinder onBind(Intent arg0) {
    return mBinder;
}
public class MyBinder extends Binder {
    LocalWordService getService() {
        return LocalWordService.this;
    }
}
public List<String> getWordList() {
    return list;
}
}

```

دو کلاس با پیاده سازی زیر تعریف نمایید. این دو کلاس در لایه‌ی XML متناظر به عنوان **BroadcastReceivers** معرفی و ثبت خواهند شد.

```

package de.vogella.android.ownservice.local;
import java.util.Calendar;
import android.app.AlarmManager;
import android.app.PendingIntent;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
public class MyScheduleReceiver extends BroadcastReceiver {
    // restart service every 30 seconds
    private static final long REPEAT_TIME = 1000 * 30;
    @Override
    public void onReceive(Context context, Intent intent) {
        AlarmManager service = (AlarmManager) context
                .getSystemService(Context.ALARM_SERVICE);
        Intent i = new Intent(context, MyStartServiceReceiver.class);
        PendingIntent pending = PendingIntent.getBroadcast(context, 0, i,
                PendingIntent.FLAG_CANCEL_CURRENT);
        Calendar cal = Calendar.getInstance();
        // start 30 seconds after boot completed
        cal.add(Calendar.SECOND, 30);
        // fetch every 30 seconds
        // InexactRepeating allows Android to optimize the energy consumption
        service.setInexactRepeating(AlarmManager.RTC_WAKEUP,
                cal.getTimeInMillis(), REPEAT_TIME, pending);
        // service.setRepeating(AlarmManager.RTC_WAKEUP, cal.getTimeInMillis(),
        // REPEAT_TIME, pending);
    }
}
package de.vogella.android.ownservice.local;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;

```

```
public class MyStartServiceReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        Intent service = new Intent(context, LocalWordService.class);
        context.startService(service);
    }
}
```

تمامی کامپوننت های نرم افزاری را با استفاده از فایل تنظیمات اپلیکیشن (AndroidManifest.xml) معرفی نمایید.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="de.vogella.android.ownservice.local"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk android:minSdkVersion="10" />
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
    <application
        android:icon="@drawable/icon"
        android:label="@string/app_name" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <service
            android:name=".LocalWordService"
            android:icon="@drawable/icon"
            android:label="@string/service_name" >
        </service>
        <receiver android:name="MyScheduleReceiver" >
            <intent-filter>
                <action android:name="android.intent.action.BOOT_COMPLETED" />
            </intent-filter>
        </receiver>
        <receiver android:name="MyStartServiceReceiver" >
        </receiver>
    </application>
</manifest>
```

محتوای فایل layout مربوط به activity را به صورت زیر ویرایش نمایید.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
```

```

    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello" />
    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="onClick"
        android:text="Update" >
    </Button>
    <ListView
        android:id="@+id/android:list"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >
    </ListView>
</LinearLayout>

```

کلاس activity خود را به صورت زیر ویرایش نمایید.

```

package de.vogella.android.ownservice.local;
import java.util.ArrayList;
import java.util.List;
import android.app.ListActivity;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.ServiceConnection;
import android.os.Bundle;
import android.os.IBinder;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.Toast;
public class MainActivity extends ListActivity {
    private LocalWordService s;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        wordList = new ArrayList<String>();
        adapter = new ArrayAdapter<String>(this,
                android.R.layout.simple_list_item_1, android.R.id.text1,
                wordList);
        setListAdapter(adapter);
    }
    @Override
    protected void onResume() {
        super.onResume();
        Intent intent= new Intent(this, LocalWordService.class);

```

```

        bindService(intent, mConnection,
            Context.BIND_AUTO_CREATE);
    }
    @Override
    protected void onPause() {
        super.onPause();
        unbindService(mConnection);
    }
    private ServiceConnection mConnection = new ServiceConnection() {
        public void onServiceConnected(ComponentName className,
            IBinder binder) {
            LocalWordService.MyBinder b = (LocalWordService.MyBinder) binder;
            s = b.getService();
            Toast.makeText(MainActivity.this, "Connected", Toast.LENGTH_SHORT)
                .show();
        }
        public void onServiceDisconnected(ComponentName className) {
            s = null;
        }
    };
    private ArrayAdapter<String> adapter;
    private List<String> wordList;
    public void onClick(View view) {
        if (s != null) {
            Toast.makeText(this, "Number of elements" + s.getWordList().size(),
                Toast.LENGTH_SHORT).show();
            wordList.clear();
            wordList.addAll(s.getWordList());
            adapter.notifyDataSetChanged();
        }
    }
}

```

اپلیکیشن را اجرا نمایید. پس از کلیک بر روی دکمه، داده ها بار دیگر از سرویس واکشی شده و لیست آیتم ها/ListView با داده های جدید بروز آوری می شوند.



## بخش دوم : زمان بندی تسك ها در اندروید با استفاده از توابع JobScheduler ای

این آموزش نحوه زمان بندی تسك ها در اندروید را با استفاده از توابع JobScheduler (API) تشریح می کند.

### 5-5- زمان بندی تسك ها

580

آدرس آموزشگاه : تهران - خیابان شریعتی - بالاتر از خیابان ملک - جنب بانک صادرات - پلاک 651 طبقه دوم - واحد 7

88146323 - 88446780 - 88146330

## 6-5-نحوه‌ی پیاده‌سازی

فرض کنید تسكی در اپلیکیشن اندرویدی خود دارید که بایستی بارها اجرا شود. در چنین شرایطی لازم است این را هم در نظر داشته باشید که سیستم اندروید ممکن است با توجه به شرایط activity ها و service ها را برای آزاد سازی منابع از حافظه حذف نماید. از این جهت نمی‌توانید TimerTask برای زمان بندی تسك ها از کلاس های ساده‌ی محیط (platform) جاوا همچون استفاده نمایید.

## 6-6-روش‌ها و ابزار مختلف زمان بندی تسك‌ها

سیستم اندروید دو ابزار برای زمان بندی اجرای تسك‌ها به شرح زیر ارائه می‌دهد:

- **AlarmManager (روش قدیمی)**
- **JobScheduler API**

اپلیکیشن‌های تحت موبایلی که امروزه برای محیط اندروید ساخته می‌شوند، برای زمان بندی و مدیریت اجرای تسك‌ها می‌بایست از توابع کتابخانه‌ای (API) JobScheduler استفاده نمایند. اپلیکیشن‌های تحت موبایل شما می‌توانند job‌ها را برای اجرا در زمان معین زمان بندی کرده، به سیستم اجازه دهند تا اجرای آن‌ها را بر اساس میزان حافظه، باتری و وضعیت اتصال مدیریت کند.

## 6-6-زمان بندی background task‌ها با استفاده از JobScheduler

## 6-6-1-توابع کتابخانه ای job scheduler

به طور کلی Job scheduler یک برنامه است که برای کنترل و مدیریت غیرحضوری برنامه هایی که در background سیستم اجرا می شوند (غالبا برای پردازش دسته ای)، مورد استفاده قرار می گیرد.

ویرایش 5.0 اندروید (API 21) توابع کتابخانه ای به نام job scheduler را در قالب کلاس JobScheduler در اختیار برنامه نویس قرار می دهد. API نام بردۀ این امکان را فراهم می آورد تا هنگامی که منابع در اختیار سیستم به اندازه ی کافی رسید، job ها به صورت دسته ای (batch) در پس زمینه به اجرا بگذارد. به بیان دیگر، توسعه دهنده می تواند با استفاده از این API تمامی کارهایی که از نظر زمان اجرا اهمیت چندانی برای کاربر ندارند را زمان بندی کند.

## 6-6-2-مزایای استفاده از JobScheduler

JobScheduler بر خلاف SyncAdapter اختصاصی یا alarm manager این قابلیت را دارد که job ها را برای پردازش دسته جمعی زمان بندی کند (آن ها را زمان بندی کرده و به هنگام رسیدن موعد تمامی کارها را دسته جمعی پردازش کند). در واقع JobScheduler با این کار (دسته بندی job ها برای پردازش دسته جمعی) میزان مصرف باتری را کاهش داده و کارایی کلی را بالا می برد.

یکی از مزایای استفاده از JobManager این است که مدیریت بارگذاری محتوا در وب را آسان ساخته و مسئله ی ناپایایی (unreliability) اینترنت را بر طرف می سازد. علاوه بر آن در صورت restart/بازآغازی، اپلیکیشن را طوری مدیریت می کند که خللی به وجود نیاید. برخی از موارد استفاده ای این JobScheduler را در زیر مشاهده می کنید:

- تسک هایی که به هنگام متصل بودن دستگاه به منبع تغذیه بایستی اجرا شوند.
- تسک هایی که برای اجرا نیاز به دسترسی به اینترنت یا Wi-Fi دارند.
- تسک هایی که ضروری نیستند یا مستقیماً با کاربر تعامل ندارند.

- تسک هایی که زمان اجرای آن ها چندان اهمیتی ندارد و از این جهت می توان آن ها را گونه ای برنامه ریزی کرد که به صورت دسته جمعی و در زمان مشخصی در آینده اجرا شوند.

### 6-6-3-نحوه‌ی ایجاد یک Job

آبجکت JobInfo حامل اطلاعات ارسالی به JobScheduler بوده و کاملاً پارامترهای مورد نیاز برای زمان بندی کارهایی که بر روی اپلیکیشن فرآخواننده بایستی انجام شود را در خود کپسوله می کند. به عبارت دیگر آبجکت JobInfo دربردارنده‌ی یک واحد کاری (unit of work) می باشد. این آبجکت مشخصات زمان بندی و کارهایی که باید انجام شود را تعیین می کند.

Job به شما این امکان را می دهد تا وضعیت دستگاه را بررسی کنید، برای مثال اینکه دستگاه بیکار است یا دسترسی به اینترنت در زمان مورد نظر امکان پذیر می باشد یا خیر و سپس بر اساس آن کارهای زمان بندی شده را انجام دهید.

جهت تعیین تنظیمات کارهای زمان بندی شده (محتوای کپسوله شده در آبجکت JobInfo) کافی است از کلاس JobInfo.Builder استفاده نمایید. می توانید تسک ها را طوری زمان بندی کنید که تحت شرایط خاصی اجرا می شود. این شرایط به شرح زیر هستند:

- زمانی که دستگاه در حال شارژ می باشد.
- زمانی که دستگاه بیکار است.
- دستگاه به اینترنت unmetered و نامحدود دسترسی دارد.
- زمانی که عملیاتی باید قبیل از تاریخ نهایی و مشخصی انجام شود.
- زمانی که کاری باید طی زمان مشخصی انجام شود برای مثال طی یک ساعت آینده.
- زمانی که عملیات باید پس از یک تأخیر جزئی به اجرا در آید، برای مثال 10 دقیقه صبر کرده و پس از آن کار زمان بندی شده را انجام دهد.

این محدودیت ها را می توان ترکیب کرد. برای مثال، می توانید یک job را به صورتی زمان بندی کنید که هر 20 دقیقه یکبار و هنگامی که دستگاه به اینترنت نامحدود وصل است اجرا شود.

یک تاریخ معین است که به عنوان محدودیت (constraint) زمانی اعمال شده و عملیات نهایتاً تا زمان سر رسید آن باید انجام شود.

به منظور پیاده سازی یک job، می بایست کلاس JobService را به ارث برده (extend) و سپس توابع onStopJob و onStartJob را پیاده سازی نمایید. چنانچه job مورد نظر به هر دلیلی ناموفق بود، لازم است مقدار بولی true را از onStopJob برگردانید تا job ناموفق دوباره انجام شود. متدهای.onStartJob در این اصلی اجرا می شود، اگر پردازش ناهمزمان را در این متدها آغاز کنید، در آن صورت می بایست مقدار بولی true و در غیر صورت false را برگردانید.

## تمرین: زمان بندی job ها با استفاده از background job

### هدف تمرین

در تمرین حاضر، یک نمونه اپلیکیشن ساده می نویسید که job ها را با فراخوانی توابع کتابخانه ای (API) JobScheduler زمان بندی می کند.

### ایجاد پروژه و فایل layout

یک پروژه جدید اندروید به نام com.vogella.android.jobscheduler بر اساس ورژن API 21 کتابخانه های اندروید ایجاد نمایید.

حال فایل layout جدید ایجاد کنید که حداقل یک کنترل دکمه در آن تعریف شده باشد. سپس متدهای onClick در لایه XML مربوطه (property) onClick را به مربوطه (property) onClick مخصوص دهید.

### 6-6-4- ایجاد JobService

یک سرویس با پیاده سازی زیر ایجاد نمایید.

```
package com.vogella.android.jobscheduler;
import java.util.LinkedList;
import android.app.job.JobInfo;
import android.app.job.JobParameters;
import android.app.job.JobScheduler;
import android.app.job.JobService;
```

```

import android.content.Context;
import android.content.Intent;
import android.os.Message;
import android.os.Messenger;
import android.os.RemoteException;
import android.util.Log;
/**
 * JobService to be scheduled by the JobScheduler.
 * Requests scheduled with the JobScheduler call the "onStartJob" method
 */
public class TestJobService extends JobService {
    private static final String TAG = "SyncService";
    @Override
    public boolean onStartJob(JobParameters params) {
        // fake work
        Log.i(TAG, "on start job: " + params.getJobId());
        return true;
    }
    @Override
    public boolean onStopJob(JobParameters params) {
        return true;
    }
}

```

**نکته:** لازم است سرویس را با استفاده از تگ `service` در لایه `manifest` (فایل تنظیمات XML) تعریف نمایید.

## 6-6-5-ساخت آزمایشی activity

پیاده سازی کلاس `activity` خود را به صورت زیر ویرایش نمایید.

```

package com.vogella.android.jobcheduler;
import android.app.Activity;
import android.app.job.JobInfo;
import android.app.job.JobScheduler;
import android.content.ComponentName;
import android.content.Context;
import android.content.res.Resources;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.RadioButton;
public class MainActivity extends Activity {
    TestJobService testService;
    private static int kJobId = 0;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

```

```
    setContentView(R.layout.activity_main);
}
public void onClick(View v) {
    ComponentName mServiceComponent = new ComponentName(this, TestJobService.class);
    JobInfo.Builder builder = new JobInfo.Builder(kJobId++, serviceComponent);
    builder.setMinimumLatency(5 * 1000); // wait at least
    builder.setOverrideDeadline(50 * 1000); // maximum delay
    builder.setRequiredNetworkType(JobInfo.NETWORK_TYPE_UNMETERED); // require unmetered
network
    builder.setRequiresDeviceIdle(true); // device should be idle
    builder.setRequiresCharging(false); // we don't care if the device is charging or not
    JobScheduler jobScheduler = getApplication().getSystemService(Context.JOB_SCHEDULER_SERVICE);
    jobScheduler.schedule(builder.build());
}
public void cancelAllJobs(View v) {
    JobScheduler tm = (JobScheduler) getSystemService(Context.JOB_SCHEDULER_SERVICE);
    tm.cancelAll();
}
```

تمرین: فراخوانی JobScheduler از داخل یک receiver/راه اندازی و

اجرای یک job از طریق service <- Receiver

هدف از تمرین

در این تمرین به صورت کاملا کاربردی خواهید آموخت چگونه می توان یک job را که داخل receiver تعریف شده، با اتفاق افتادن رخداد خاصی (که receiver به آن گوش می دهد) فعال نمایید.

## receiver-6-6-6 ایجاد

Receiver را به صورت زیر پیاده سازی کنید.

```
package com.example.android.rssreader;
import android.app.job.JobInfo;
import android.app.job.JobScheduler;
import android.content.BroadcastReceiver;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
public class BootReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent)
```

```

        ComponentName serviceComponent = new ComponentName(context, MyJobService.class);
        JobInfo.Builder builder = new JobInfo.Builder(0, serviceComponent);
        builder.setMinimumLatency(5 * 1000); // wait at least
        builder.setOverrideDeadline(50 * 1000); // maximum delay
        builder.setRequiredNetworkType(JobInfo.NETWORK_TYPE_UNMETERED); // require unmetered network
        builder.setRequiresDeviceIdle(true); // device should be idle
        builder.setRequiresCharging(false); // we don't care if the device is charging or not
        JobScheduler jobScheduler = context.getSystemService(JobScheduler.class);
        jobScheduler.schedule(builder.build());
    }
}

```

## Job-6-ایجاد 7

```

package com.example.android.rssreader;
import android.app.job.JobParameters;
import android.app.job.JobService;
import android.content.Intent;
/**
 * Created by vogella on 30.06.16.
 */
public class MyJobService extends JobService {
    @Override
    public boolean onStartJob(JobParameters params) {
        String url = params.getExtras().getString(RssApplication.URL);
        Intent i = new Intent(this, RssDownloadService.class); // starts the RssDownloadService service
        i.putExtra(RssApplication.URL, url); // some extra data for the service
        startService(i);
        return false; // true if we're not done yet
    }
    @Override
    public boolean onStopJob(JobParameters params) {
        // true if we'd like to be rescheduled
        return true;
    }
}

```

می توانید receiver (گوش دهنده به رخدادها) را برای event خاصی همچون رخداد boot و بالا آمدن سیستم ثبت و معرفی نمایید. receiver مذکور مربوطه را از اتفاق مورد نظر باخبر نموده که آن خود job ای که سرویس اختصاصی را اجرا می کند، فعال می نماید.

## بخش سوم :

# آموزش استفاده از Broadcast receiver (گوش فراخوان برای رخدادها) در اندروید

این آموزش به شرح نحوه ی ایجاد و استفاده از کامپوننت نرم افزاری broadcast receiver در اندروید را شرح می دهد.



یک کامپوننت نرم افزاری در اندروید است که به شما این امکان را می دهد تا به event های مختلف گوش فرا دهید. به عبارت دیگر broadcast receiver یک نوع گوش فراخوان به رخدادها است.

زمانی که رخداد یا event مورد نظر رخ می دهد، runtime سیستم اندروید تمامی receiver هایی که برای آن رخداد ثبت شده و به آن گوش می دهند را با خبر می کند.

برای مثال می توان به اپلیکیشن هایی اشاره کرد که به رخداد سیستمی ACTION\_BOOT\_COMPLETED گوش داده و زمانی که سیستم اندروید پروسه ی boot را به اتمام می رساند، اپلیکیشن را از این اتفاق با خبر می سازد.

## 6-7-2-پیاده سازی receiver

Receiver را بایستی در لایه XML، درون فایل تنظیمات AndroidManifest.xml تعریف نمایید.

یک receiver را می‌توان به صورت فوق تعریف کرد که روش static نام دارد یا می‌توان آن را به صورت dynamic در کلاس های جاوا با فراخوانی متده Context.registerReceiver() اعلان نمایید.

برای تعریف receiver لازم است از کلاس BroadcastReceiver ارث بری نمایید.

اگر event ای که broadcast receiver برای آن ثبت شده رخ دهد، سیستم اندروید متده onReceive() را از receiver را صدا می‌زند.

## 6-7-3-broadcast receiver یا چرخه‌ی حیات کامپوننت

پس از اینکه اجرای متده onReceive() از کلاس receiver به پایان رسید، سیستم اندروید این اجازه را دارد که receiver را برای استفاده مجدد در آینده (recycle) بازیافت نماید.

## 6-7-4-پردازش ناهمزمان (async processing)

پیش از API 11 توسعه دهنده‌ان امکان اجرای هیچ گونه عملیات ناهمزمانی را در متده onReceive() نداشتند چرا که به محض به اتمام رسیدن اجرای متده onReceive() سیستم اندروید می‌توانست آن کامپوننت را بازیافت نماید. چنانچه عملیات طولانی برای اجرا وجود داشته باشد، در این شرایط بهتر است بجای broadcast receiver، یک کامپوننت سرویس را فراخوانی نمایید.

از ورژن 11 کتابخانه‌های اندروید، شما می‌توانید متده goAsync() را برای اجرای عملیات ناهمزمان فراخوانی نمایید. این متده در خروجی آبجکتی از جنس PendingResult برمی‌گرداند. لازم به ذکر است که تا زمانی که متده PendingResult.finish() را بر روی این آبجکت فراخوانی نکرده‌اید، سیستم اندروید receiver را همچنان فعال در نظر می‌گیرد. از این طریق شما می‌توانید در receiver، عملیات ناهمزمان راه اندازی نمایید. بلاfaciale پس از اینکه thread پردازش خود را به

پایان می رساند، تسك (میزبان) متدها finish() را صدا زده و به سیستم اندروید اعلان می کند که کامپوننت مورد بحث اکنون قابل بازیافت (استفاده ای مجدد و recycle) می باشد.

## 6-7-5- محدودیت هایی در تعریف و استفاده از broadcast receiver

از ویرایش 3.1 به بعد اندروید، چنانچه اپلیکیشن مورد نظر را کاربر راه اندازی نکرده یا اینکه آن را به صورت صریح از طریق منوی اندروید (Manage > Application) خاتمه داده باشد، آنگاه سیستم به صورت پیش فرض تمامی receiver ها را از دریافت intent ها محروم خواهد نمود.

این یک امکان امنیتی است که به واسطه‌ی آن کاربر اطمینان دارد تنها اپلیکیشن هایی که خود به صورت صریح اجرا کرده قادر به دریافت broadcast intent ها خواهند بود.

**نکته:** این بدین معنی نیست که کاربر بایستی اپلیکیشن را مجددا پس از reboot اجرا کند. بلکه سیستم اندروید به یاد دارد که کاربر قبل از آن را یکبار اجرا کرده است. بنابراین چنانچه کاربر اپلیکیشن را اجرا کرده باشد (آن را به صورت صریح نبسته باشد) سیستم اندروید پس از reboot به خاطر می آورد که برنامه‌ی مورد بحث یکبار قبل اجرا شده و خود اقدام به اجرای مجدد آن می نماید.

## 6-7-6- ارسال broadcast به اپلیکیشن به منظور تست

دستورات زیر را در پنجره‌ی فرمان adb (adb command line tool) درج نمایید. اسم کلاس و پکیج که در دستور زیر مشاهده می کنید، می بایست قبل از لایه XML، داخل فایل manifest تعریف شده باشد. شما بایستی intent و داده های کپسوله در آن را به کامپوننت مورد نظر ارسال نمایید. حال زمانی که broadcast سراسری به نام ACTION\_BOOT\_COMPLETED را به کامپوننت نرم افزاری مدنظر ارسال می کنید، چندین اتفاق متعدد در سیستم اندروید رخ می دهد.

```
# trigger a broadcast and deliver it to a component
adb shell am activity/service/broadcast -a ACTION -c CATEGORY -n NAME
```

```
# for example (this goes into one line)
adb shell am broadcast -a
android.intent.action.BOOT_COMPLETED -c android.intent.category.HOME -n
package_name/class_name
```

## Pending Intent 6-7- شرح مفهوم

یک آبجکت متشکل از درخواست عملیات دلخواه (راه اندازی یک activity، اجرای سرویس یا ارسال broadcast) اطلاعات و جزئیات عملیات (کپسوله شده در یک intent) و در نهایت یک context می باشد. نمونه ی مشتق شده از کلاس PendingIntent را می توان به سایر اپلیکیشن های اندرویدی واگذار کرد تا حتی پس از بسته شدن برنامه ی میزبان یا ایجاد کننده ی pending intent، بتواند عملیاتی را به انجام برساند. زمانی که توسعه دهنده pending intent را به اپلیکیشن دیگری می فرستد، در عمل به آن برنامه اجازه می دهد تا مجوزهای اپلیکیشن ایجاد کننده ی pending intent را جهت انجام عملیات خاص داشته باشد. این بر خلاف intent های متعارف است که اپلیکیشن های دریافت کننده ی آن می بایست برای انجام عملیات از مجوزهای خود استفاده کنند.

به عبارت دیگر pending intent یک token است که برنامه نویس به اپلیکیشن دیگری مانند alarm manger یا هر اپلیکیشن دیگری می دهد. با ارسال این نوع intent به اپلیکیشن دیگر، در واقع توسعه دهنده به اپلیکیشن مقصد این اجازه را می دهد تا مجوزهای اپلیکیشن فرستنده را جهت اجرای کد و عملیات از پیش تعریف شده داشته باشد.

برای اجرای یک broadcast از طریق pending intent، ابتدا متدهای getBroadcast() و getPendingIntent() را فراخوانی نموده و آبجکتی از کلاس PendingIntent را دریافت نمایید. حال جهت اجرا activity از طریق PendingIntent.getActivity() و pending intent مورد نظر را به عنوان خروجی از این متدهای دریافت می کنید.

## Broadcast-8-6 رخدادهای سیستمی

کلاس Intent بسیاری از رخدادهای سیستمی را در قالب فیلدهای ایستا و ثابت (final و static) به صورت آماده (از قبل تعریف شده) در خود کپسوله می‌کند. سایر کلاس‌های سیستم اندروید نیز رخدادهای سیستمی دیگری را تعریف می‌کنند. به عنوان مثال می‌توان به TelephoneManager برای تغییر وضعیت گوشی اشاره کرد.

جدول زیر تعدادی رخداد سیستمی بسیار مهم را همراه با شرح کاربرد ذکر می‌کند.

جدول 1- رخدادهای سیستمی

Event	شرح کاربرد
Intent.ACTION_BOOT_COMPLETED	سیستم با پایان رسیده است. برای اجرای نیاز به مجوز android.permission.RECEIVE_BOOT_COMPLETED دارد.
Intent.ACTION_POWER_CONNECTED	دستگاه به منبع تغذیه وصل شده است.
Intent.ACTION_POWER_DISCONNECTED	دستگاه دیگر به منبع تغذیه وصل نیست.
Intent.ACTION_BATTERY_LOW	زمانی اجرا می‌شود که باتری دستگاه رو به اتمام باشد. معمولاً برای کاستن از تعداد های activity اپلیکیشنی که میزان مصرف باتری آن بالا است، مورد استفاده قرار می‌گیرد.
Intent.ACTION_BATTERY_OKAY	باتری دستگاه بار دیگر در وضعیت مناسبی قرار دارد.

## Receiver-9-6 اجرا و راه اندازی سرویس به صورت خودکار

یکی از رفتارهایی که زیاد مورد نیاز بوده و پیاده سازی می‌شود، فراخوانی و اجرای یک سرویس به صورت خودکار پس از reboot سیستم است. این کار می‌تواند برای مثال جهت همگاهنگ سازی داده‌ها صورت گیرد. برای این منظور ابتدا یک receiver (گوش فراخوان به رخداد) برای

لازم است مجوز android.intent.action.BOOT\_COMPLETED تعریف می کنید. لازم است android.permission.RECEIVE\_BOOT\_COMPLETED را نیز در فایل تنظیمات برای اجرای اعمالیات مذبور به اپلیکیشن اعطای نمایید.

مثال زیر نحوه تعریف رخداد BOOT\_COMPLETED در فایل تنظیمات (Manifest) را به نمایش می گذارد.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="de.vogella.android.ownservice.local"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk android:minSdkVersion="10" />
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
    <application
        android:icon="@drawable/icon"
        android:label="@string/app_name" >
        <activity
            android:name=".ServiceConsumerActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <receiver android:name="MyScheduleReceiver" >
            <intent-filter>
                <action android:name="android.intent.action.BOOT_COMPLETED" />
            </intent-filter>
        </receiver>
        <receiver android:name="MyStartServiceReceiver" >
        </receiver>
    </application>
</manifest>
```

به دنبال تعریف رخداد فوق در لایه XML، کامپوننت receiver می بایست سرویس را به وسیله `Intent` کد زیر به صورت خودکار و با بالا آمدن سیستم راه اندازی کند.

```
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
public class MyReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
```

```
// assumes WordService is a registered service  
Intent intent = new Intent(context, WordService.class);  
context.startService(intent);  
}  
}
```

**توجه:** چنانچه اپلیکیشن در کارت SD (حافظه خارجی) مستقر باشد، در آن صورت پس از رخداد android.intent.action.BOOT\_COMPLETED، اپلیکیشن مورد نظر دیگر در دسترس نخواهد بود. در این شرایط می بایست به رخداد گوش داده و android.intent.action.ACTION\_EXTERNAL\_APPLICATIONS\_AVAILABLE را داخل فایل تنظیمات اپلیکیشن برای آن ثبت نمایید. Receiver

**نکته:** به خاطر داشته باشید از ورژن API 11 کتابخانه های اندروید، برای اینکه اپلیکیشن بتواند رخداد android.intent.action.BOOT\_COMPLETED events را دریافت کند، کاربر می بایست حداقل یکبار قبل اپلیکیشن را اجرا کرده باشد.

تمرین: ثبت یک RECEIVER جهت گوش فراخوانی به تماس های دریافتی

هدف از تمرین

در تمرین حاضر یک broadcast receiver تعریف می کنید که به رخدادهای مربوط به وضعیت دستگاه تلفن گوش می دهد. اگر دستگاه یک تماس دریافتی داشته باشد، در آن صورت receiver از رخداد آگاه شده و یک پیغام مربوطه ثبت (log) می نماید.

ساخت پروژه

یک پروژه‌ی جدید به نام de.vogella.android.receiver.phone ایجاد نموده و سپس یک جدید تعریف نمایید.

**نکته:** به یاد داشته باشید که receiver تنها به شرطی که کاربر قبل آن را یکبار اجرا کرده باشد، صدا خورده می شود. برای این منظور لازم است یک activity تعریف نمایید.

## 6-9-1-پیاده سازی receiver برای گوش فرآخوانی به رخدادهای مربوطه به وضعیت تلفن و تماس ها

کلاس MyPhoneReceiver را ایجاد نمایید.

```
package de.vogella.android.receiver.phone;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.telephony.TelephonyManager;
import android.util.Log;
public class MyPhoneReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        Bundle extras = intent.getExtras();
        if (extras != null) {
            String state = extras.getString(TelephonyManager.EXTRA_STATE);
            Log.w("MY_DEBUG_TAG", state);
            if (state.equals(TelephonyManager.EXTRA_STATE_RINGING)) {
                String phoneNumber = extras
                    .getString(TelephonyManager.EXTRA_INCOMING_NUMBER);
                Log.w("MY_DEBUG_TAG", phoneNumber);
            }
        }
    }
}
```

6-9-2-اعطا مجوز گوش فرآخوانی به تغییرات مربوط به وضعیت در receiver  
 مجوز android.permission.READ\_PHONE\_STATE را به فایل تنظیمات (manifest) اضافه نموده تا اپلیکیشن قابلیت گوش فرآخوانی به تغییرات در وضعیت (از طریق receiver) را داشته باشد. سپس receiver مورد نظر را در لایه XML (داخل فایل Manifest) تعریف نمایید. فایل manifest پس از وارد نمودن تغییرات فوق بایستی ظاهری مشابه زیر داشته باشد.

## تست اپلیکیشن

اپلیکیشن خود را نصب نموده و سپس از طریق ADV (android device monitor) یک تماس تلفنی به دستگاه را شبیه سازی نمایید. مطمئن شوید که receiver فراخوانی شده و یک پیغام در LogCat view ثبت می شود.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="de.vogella.android.receiver.phone"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk android:minSdkVersion="15" />
    <uses-permission android:name="android.permission.READ_PHONE_STATE" >
    </uses-permission>
    <application
        android:icon="@drawable/icon"
        android:label="@string/app_name" >
        <activity
            android:name=".MainActivity"
            android:label="@string/title_activity_main" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <receiver android:name="MyPhoneReceiver" >
            <intent-filter>
                <action android:name="android.intent.action.PHONE_STATE" >
                </action>
            </intent-filter>
        </receiver>
    </application>
</manifest>
```

## تمرین: receiver و سرویس های سیستم

### هدف از تمرین

در این تمرین یک receiver را از طریق سرویس درون ساخته‌ی محیط اندروید زمان بندی خواهید کرد. پس از اینکه receiver فراخوانی می شود، این کامپوننت نرم افزاری با استفاده از Android vibrator manager (popup message) و یک پیغام (Toast) کاربر را از تغییرات مربوطه آگاه می سازد.

## پیاده سازی پروژه

یک پروژه و activity جدید به ترتیب به نام های de.vogella.android.alarm و AlarmActivity ایجاد نمایید.

سپس فایل یک layout با محتوای زیر ایجاد کنید.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <EditText
        android:id="@+id/time"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:hint="Number of seconds"
        android:inputType="numberDecimal" >
    </EditText>
    <Button
        android:id="@+id	ok"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="startAlert"
        android:text="Start Counter" >
    </Button>
</LinearLayout>
```

کلاس vibrator را به صورت زیر پیاده سازی نمایید. این کلاس سرویس صدا می زند.

```
package de.vogella.android.alarm;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Vibrator;
import android.widget.Toast;
public class MyBroadcastReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        Toast.makeText(context, "Don't panik but your time is up!!!!.", Toast.LENGTH_LONG).show();
```

```

    // Vibrate the mobile phone
    Vibrator vibrator = (Vibrator) context.getSystemService(Context.VIBRATOR_SERVICE);
    vibrator.vibrate(2000);
}
}

```

این کلاس را در لایه ی XML، داخل فایل AndroidManifest.xml اعلان نموده و اجازه ی vibrate برای دستگاه را به اپلیکیشن اعطای نمایید.



```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="de.vogella.android.alarm"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk android:minSdkVersion="15" />
    <uses-permission android:name="android.permission.VIBRATE" >
    </uses-permission>
    <application
        android:icon="@drawable/icon"
        android:label="@string/app_name" >
        <activity
            android:name=".AlarmActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <receiver android:name="MyBroadcastReceiver" >
        </receiver>
    </application>
</manifest>

```

بدنه ی کلاس AlarmActivity خود را به صورت زیر ویرایش نمایید. این activity داخل خود یک آبجکت intent تعریف نموده و به وسیله ی آن receiver را فراخوانی می کند، سپس اطلاعات این را به سرویس alarm manager معرفی/ارسال می نماید.

```

package de.vogella.android.alarm;
import android.app.Activity;
import android.app.AlarmManager;
import android.app.PendingIntent;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

```

```

public class AlarmActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
    public void startAlert(View view) {
        EditText text = (EditText) findViewById(R.id.time);
        int i = Integer.parseInt(text.getText().toString());
        Intent intent = new Intent(this, MyBroadcastReceiver.class);
        PendingIntent pendingIntent = PendingIntent.getBroadcast(
            this.getApplicationContext(), 234324243, intent, 0);
        AlarmManager alarmManager = (AlarmManager) getSystemService(ALARM_SERVICE);
        alarmManager.set(AlarmManager.RTC_WAKEUP, System.currentTimeMillis()
            + (i * 1000), pendingIntent);
        Toast.makeText(this, "Alarm set in " + i + " seconds",
            Toast.LENGTH_LONG).show();
    }
}

```

## تست اپلیکیشن

اپلیکیشن خود را روی دستگاه تست نمایید. پس از تنظیم زمان، alarm را فراخوانی کنید. اکنون با گذشت تعداد ثانیه های تعیین شده، یک پیغام Toast به نمایش در می آید. لازم به ذکر است که قابلیت vibration در شبیه ساز اندروید کار نمی کند.



## 6-10-تعریف dynamic broadcast receiver به صورت (در زمان اجرای برنامه و به وسیله کدهای جاوا)

### 6-10-1 Receiver هایی که به صورت dynamic اعلان شده

همان طور که قبلا گفته شد می توانید receiver را در فایل manifest اندروید ثبت نمایید که از آن تحت عنوان ثبت به صورت static یاد می شود. در این بخش ایجاد و حذف receiver در زمان

600

آدرس آموزشگاه : تهران - خیابان شریعتی - بالاتر از خیابان ملک - جنب بانک صادرات - پلاک 651 طبقه دوم - واحد 7

88146323 - 88446780 - 88146330

اجرای برنامه به وسیله ی متدهای Context.registerReceiver() و Context.unregisterReceiver() را شرح خواهیم داد.

**توجه:** لازم است receiver ای که به صورت dynamic تعریف کرده اید را با فراخوانی متدهای Context.unregisterReceiver() حذف (unregister) نمایید. در صورت عدم فراخوانی متدهای Context.unregisterReceiver() مذکور، سیستم اندروید پیغام خطای leaked broadcast receiver را صادر می کند. بنابراین اگر receiver را داخل بدنه ی متدهای onResume() و onPause() کلاس activity خود اعلان کرده باشید، بایستی آن را داخل (unregister) حذف (onPause()) نمایید.

## 2-10-6-استفاده از package manager جهت غیرفعال سازی receiver های static

می توانید با استفاده از کلاس PackageManager به راحتی receiver های تعریف شده در لایه XML (AndroidManifest.xml) را فعال/غیرفعال نمایید.

```
ComponentName receiver = new ComponentName(context, myReceiver.class);
PackageManager pm = context.getPackageManager();
pm.setComponentEnabledSetting(receiver,
    PackageManager.COMPONENT_ENABLED_STATE_ENABLED,
    PackageManager.DONT_KILL_APP);
```

## 3-10-6-Intent های ماندگار (Sticky broadcast intent)

یک intent معمولاً پس از اینکه ارسال شده و سیستم اطلاعات درون آن را پردازش کرد، دیگر در دسترس نبوده و از حافظه پاک خواهد شد. چنانچه مایلید آبجکت مزبور پس از پردازش توسط سیستم همچنان باقی ماند و به اصطلاح ماندگار باشد، کافی است متدهای sendStickyBroadcast(Intent) را فراخوانی نمایید. در صورت فراخوانی این متدها، intent ماندگار بوده و پس از اتمام broadcast، در حافظه باقی می ماند.

سیستم اندروید برای حفظ برخی از اطلاعات سیستمی از sticky broadcast/intent استفاده می کند. برای مثال می توان به وضعیت باتری اشاره کرد که سیستم آن را ارسال نموده و پس از ارسال تا مدتی باقی می ماند.

مثال:

```
// Register for the battery changed event
IntentFilter filter = new IntentFilter(Intent.ACTION_BATTERY_CHANGED);
// Intent is sticky so using null as receiver works fine
// return value contains the status
Intent batteryStatus = this.registerReceiver(null, filter);
// Are we charging / charged?
int status = batteryStatus.getIntExtra(BatteryManager.EXTRA_STATUS, -1);
boolean isCharging = status == BatteryManager.BATTERY_STATUS_CHARGING
    || status == BatteryManager.BATTERY_STATUS_FULL;
boolean isFull = status == BatteryManager.BATTERY_STATUS_FULL;
// How are we charging?
int chargePlug = batteryStatus.getIntExtra(BatteryManager.EXTRA_PLUGGED, -1);
boolean usbCharge = chargePlug == BatteryManager.BATTERY_PLUGGED_USB;
boolean acCharge = chargePlug == BatteryManager.BATTERY_PLUGGED_AC;
```

می توانید داده ها را از مقدار بازگشتی متده registerReceiver(BroadcastReceiver, IntentFilter) بازیابی نمایید. این روش برای BroadcastReceiver که حامل مقدار null می باشد نیز قابل استفاده است.

در دیگر جنبه ها، این متده رفتاری مشابه متده sendBroadcast(Intent) دارد.

استفاده از sticky broadcast ها لازمه ای تنظیم مجوزهای خاصی در لایه ای XML می باشد.

## بخش چهارم : آموزشگاه تحلیلکردن و داده های notification manager در اندروید/پیاده سازی notification

این مبحث به نحوه ای استفاده از notification manager در اندروید می پردازد. پروژه ای آموزش حاضر در محیط برنامه نویسی Android studio نوشته شده و مبتنی بر ویرایش 5.0 سیستم عامل اندروید می باشد.

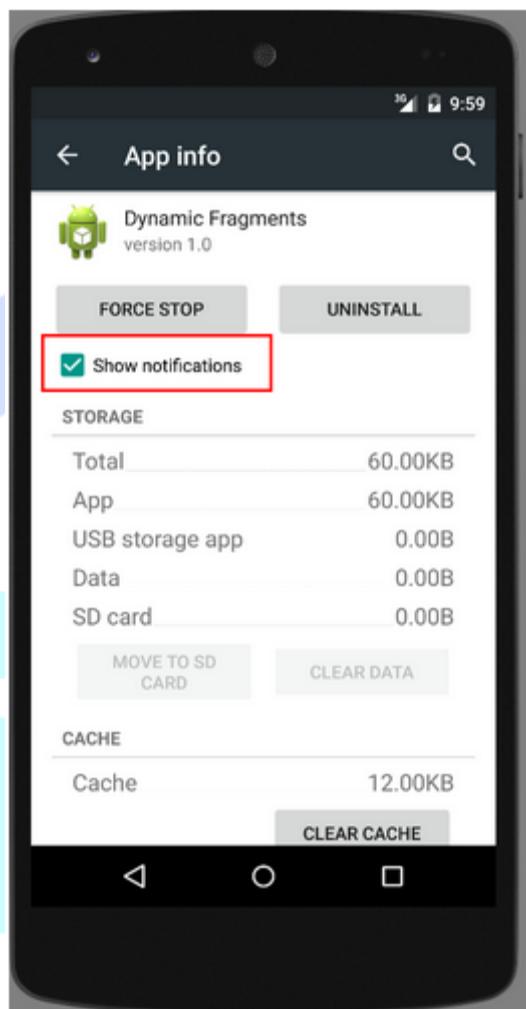
## Notification manager-11-6

### notification manager مفهوم 11-6-شرح

در سیستم عامل اندروید توسعه دهنده قادر است با استفاده از notification هشدارهای سیستمی یا مربوط به اپلیکیشن را به کاربر نمایش دهد.

اندروید به شما این امکان را می دهد تا notification ها را داخل titlebar اپلیکیشن خود نمایش دهید. کاربر می تواند بر روی notification تپ کرده، آن را باز نماید. سپس با انتخاب notification یک activity و صفحه ی جدید را به اجرا در بیاورد.

از آنجایی که notification ها کیفیت تجربه ی کاربری را پایین آورده و کمی آزار دهنده هستند، کاربر این اجازه را دارد که notification های اپلیکیشن مربوطه را غیرفعال نماید. برای نیل به این هدف، کافی است کاربر به بخش تنظیمات Settings اپلیکیشن در دستگاه اندروید مراجعه کند. پس از انتخاب گزینه Apps در Settings، کاربر اپلیکیشنی که می خواهد notification های آن را غیر فعال کند، انتخاب نموده و تیک چک باکس show notifications اپلیکیشن مرتبط را برمی دارد.



## ۱۱-۶- تنظیم و مقدار آندهی notification ها

برای ساخت و نمایش notification در اندروید از کلاس Notification استفاده می شود.

نحوه ی ساخت notification در اندروید: به منظور مقداردهی و تنظیم notification از کلاس NotificationManager استفاده می شود. این کلاس با فراخوانی متد () getSystemService() از کلاس Context قابل بازیابی می باشد.

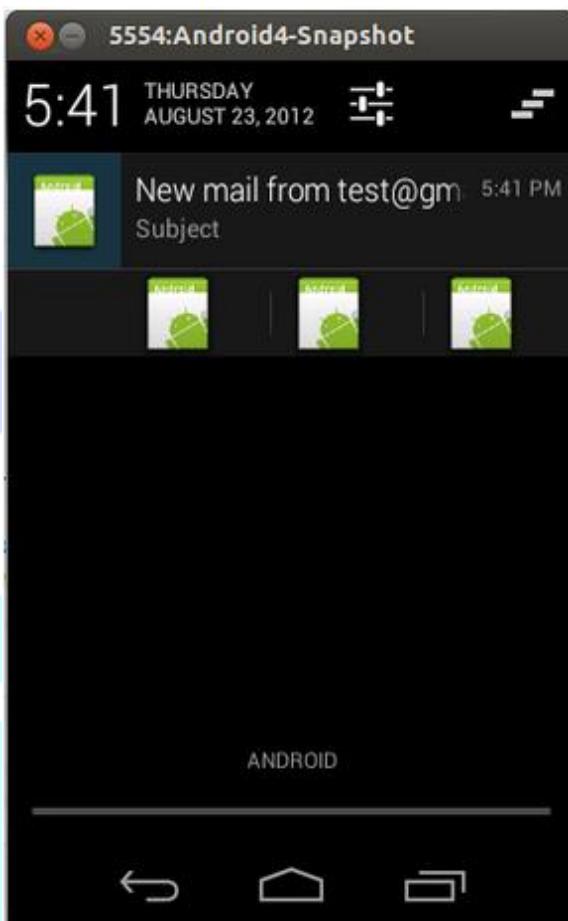
```
NotificationManager notificationManager = (NotificationManager)
        getSystemService(NOTIFICATION_SERVICE);
```

یک interface `Notification` جهت ساخت و مقداردهی آبجکت های لازم از کلاس `Notification.Builder` در اختیار توسعه دهنده قرار می دهد.

می توانید با استفاده از آبجکت `notification`, عملیاتی که پس از انتخاب `PendingIntent` باید اجرا شود را تعیین نمایید.

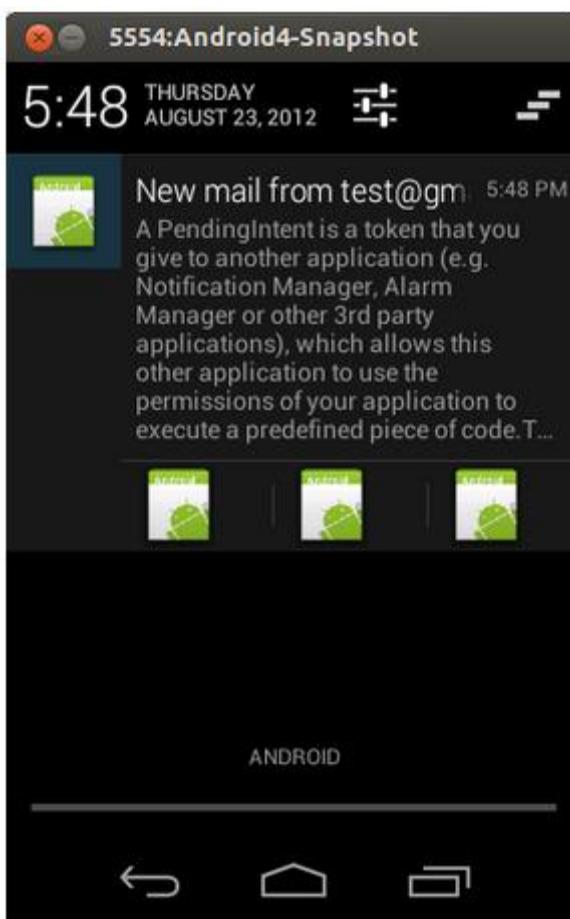
به شما این امکان را می دهد که تا 3 کنترل دکمه با رفتار (action) دلخواه به `notification` اضافه نمایید.

```
// prepare intent which is triggered if the
// notification is selected
Intent intent = new Intent(this, NotificationReceiver.class);
// use System.currentTimeMillis() to have a unique ID for the pending intent
PendingIntent plntent = PendingIntent.getActivity(this, (int) System.currentTimeMillis(), intent, 0);
// build notification
// the addAction re-use the same intent to keep the example short
Notification n = new Notification.Builder(this)
    .setContentTitle("New mail from " + "test@gmail.com")
    .setContentText("Subject")
    .setSmallIcon(R.drawable.icon)
    .setContentIntent(plntent)
    .setAutoCancel(true)
    .addAction(R.drawable.icon, "Call", plntent)
    .addAction(R.drawable.icon, "More", plntent)
    .addAction(R.drawable.icon, "And more", plntent).build();
NotificationManager notificationManager =
    (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
notificationManager.notify(0, n);
```



ویرایش 4.1 اندروید به کاربران این امکان را می‌دهد تا notification‌ها را باز کرده و جزئیات بیشتری را مشاهده نمایند. در واقع از ورژن ذکر شده به بعد علاوه بر نمای (view) معمولی notification، نمای بزرگی نیز تعبیه شده که به هنگام باز شدن notification فراخوانی می‌شود. برای طراحی ظاهر notification، سه style قابل گزینش می‌باشد: 1. big picture style .2 big text style .3 text style از کل نمایشگر را به notification اختصاص می‌دهد.

```
String longText = "...";
Notification noti = new Notification.Builder(this)
.....
.setStyle(new Notification.BigTextStyle().bigText(longText))
```



### ۱۱-۶-لغو کردن notification ها

کاربر می تواند تمامی notification ها را لغو کند. چنانچه توسعه دهنده notification را بر روی `cancel()` متد تنظیم کرده باشد، آنگاه به هنگام انتخاب کاربر، notification از نمایشگر حذف می شود.

می توانید ID یا شناسه ی منحصر بفرد `Notification` را به عنوان پارامتر به متد `cancel()` پاس داده و آن را بر روی آبجکت `NotificationManager` فراخوانی نمایید. با فراخوانی متد `cancelAll()` تمامی notification هایی که قبلا صادر کرده بودید، حذف می شوند.

## تمرین: NotificationManager

607

آدرس آموزشگاه: تهران - خیابان شریعتی - بالاتر از خیابان ملک - جنب بانک صادرات - پلاک ۶۵۱ طبقه دوم - واحد ۷

88146323 - 88446780 - 88146330

یک پروژه و activity جدید به ترتیب به نام های de.vogella.android.notificationmanager و CreateNotificationActivity ایجاد نمایید. برای تنظیم ظاهر خود بایستی فایل main.xml را فراخوانی کند.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <Button
        android:id="@+id/button1"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:onClick="createNotification"
        android:text="Create Notification" >
    </Button>
</LinearLayout>
```

فایل result.xml را با محتوای زیر ایجاد نمایید.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="This is the result activity opened from the notification" >
    </TextView>
</LinearLayout>
```

اکنون کلاس activity دیگری به نام NotificationReceiverActivity ایجاد کرده و بدنه‌ی آن را به صورت زیر تنظیم نمایید. به یاد داشته باشید که activity مربوطه را حتماً داخل فایل تنظیمات AndroidManifest.mf تعریف کنید.

```
package de.vogella.android.notificationmanager;
import android.app.Activity;
import android.os.Bundle;
public class NotificationReceiverActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.result);
```

سیاده سازی کلاس CreateNotificationActivity را به صورت زیر ویرایش نمایید.

```
package de.vogella.android.notificationmanager;
import android.app.Activity;
import android.app.Notification;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
public class CreateNotificationActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
    public void createNotification(View view) {
        // Prepare intent which is triggered if the
        // notification is selected
        Intent intent = new Intent(this, NotificationReceiverActivity.class);
        PendingIntent plntent = PendingIntent.getActivity(this, (int) System.currentTimeMillis(), intent, 0);
        // Build notification
        // Actions are just fake
        Notification noti = new Notification.Builder(this)
                .setContentTitle("New mail from " + "test@gmail.com")
                .setContentText("Subject").setSmallIcon(R.drawable.icon)
                .setContentIntent(plntent)
                .addAction(R.drawable.icon, "Call", plntent)
                .addAction(R.drawable.icon, "More", plntent)
                .addAction(R.drawable.icon, "And more", plntent).build();
        NotificationManager notificationManager = (NotificationManager)
getSystemService(NOTIFICATION_SERVICE);
        // hide the notification after its selected
        noti.flags |= Notification.FLAG_AUTO_CANCEL;
        notificationManager.notify(0, noti);
    }
}
```

اپلیکیشن خود را اجرا کرده و کنترل دکمه را فشار دهید. یک notification جدید ایجاد شده که با انتخاب آن، activity و پنجره‌ی جدید در نمایشگر باز می‌شود.

بخش پنجم:

# گیری از داده ها در اندروید/تهیه ی نسخه ی Backup

## پشتیبان از داده ها در اندروید

آموزش حاضر به شرح توابع کتابخانه ای Android data backup API پرداخته و نحوه ی استفاده از آن برای بازگردانی داده های مربوط به تنظیمات و Configuration اپلیکیشن را توضیح می دهد.

### Backup-12-6 در اندروید

#### 12-6-هدف از تهیه ی نسخه ی پشتیبان از داده ها

سرویس backup گیری اندروید به توسعه دهنده این امکان را می دهد تا داده های ماندگار اپلیکیشن را در حافظه ی راه دور cloud ذخیره نماید. بدین وسیله زمانی که کاربر دستگاه اندروید خود را عوض می کند یا اپلیکیشنی را مجدداً نصب می نماید و یا factory reset گوشی را فعال می کند، داده های اپلیکیشن و اطلاعات مربوط به تنظیمات همگی به راحتی قابل بازگردانی خواهد بود.

البته تمامی دستگاه های اندروید لزوماً قابلیت data backup را ندارند. همچنین حافظه ی cloud ممکن است از جانب تولید کننده ی دستگاه مورد نظر به صورت اختصاصی تنظیم شده باشد. از این رو تیم توسعه ی محیط اندروید به هیچ وجه امنیت و سلامت کامل داده ها را به هنگام backup و بازگردانی تضمین نمی کند.

چنانچه سرویس در دستگاه مذکور موجود نباشد، متعاقباً سرویس backup فراخوانی نمی شود. از این رو بهتر است ابتدا سرویس را پیاده سازی نموده و سپس آن را بر روی دستگاه دلخواه اجرا نمایید.

برای استفاده از سرویس backup گیری، ابتدا لازم است اسم پکیج اپلیکیشن خود را در آدرس اینترنتی <https://developer.android.com/backup/signup.html> ثبت نمایید. پس از

معرفی کردن اسم پکیج اپلیکیشن خود، این سایت یک کلید backup در اختیار شما قرار می دهد که برای تهیه ی نسخه ی پشتیبان از داده های اپلیکیشن مورد نظر در آینده از آن استفاده خواهد نمود.

## 6-12-2-تهیه ی نسخه ی پشتیبان از shared preferences (داده های

کوچک همچون اطلاعات مربوط به تنظیمات اپلیکیشن) و فایل ها برای ذخیره ی دائمی داده ها و بازگردانی آن ها، ابتدا یک کلاس تعریف کنید که توابع و فیلدهای کلاس BackupAgent را به ارث می برد.

ساده ترین راه برای پیاده سازی backup استفاده از SharedPreferencesBackupHelper ویژه ی backup گیری از داده های کوچک همچون اطلاعات مربوط به تنظیمات اپلیکیشن و استفاده از FileBackupHelper ویژه ی تهیه ی نسخه ی پشتیبان از داده های مستقر در حافظه ی داخلی سیستم می باشد. دو کلاس کمکی (helper) نام بردہ به صورت خودکار تمامی فایل های ثبت شده را ذخیره کرده و بازگردانی می نماید.

## 6-12-3-تهیه ی Backup کلی

به منظور تهیه ی backup پیچیده و کلی تر، می توانید خود کلاس BackupAgent و متدهای restore/save آن را به صورت مستقیم پیاده سازی نمایید. برای مطالعه ی جزئیات بیشتر می توانید به آدرس <http://developer.android.com/guide/topics/data/backup.html> مراجعه نمایید.

**تمرین: پیاده سازی backup گیری از داده های مربوط به تنظیمات اپلیکیشن**

**پیاده سازی پروژه**

اپلیکیشنی که کد آن را در زیر مشاهده می کنید از پکیج com.vogella.android.databackup است. بالای XML استفاده می کند. این پکیج در گوگل سرویس ثبت شده است.

به AndroidManifest.xml در فایل تنظیمات اپلیکیشن registration key و Backup agenda صورت زیر درج می شود.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.vogella.android.databackup"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk
        android:minSdkVersion="17"
        android:targetSdkVersion="17" />
    <application
        android:allowBackup="true"
        android:backupAgent="MyBackupAgent"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.vogella.android.databackup.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <meta-data
            android:name="com.google.android.backup.api_key"
            android:value="AEdPqrEAAAII4SfiyaQncNamlUH0NboU3tzOjXGztXLv2LZkEw" />
    </application>
</manifest>
```

مربوطه را می توان به صورت زیر پیاده سازی کرد.

```
package com.vogella.android.databackup;
import android.app.backup.BackupAgentHelper;
import android.app.backup.SharedPreferencesBackupHelper;
public class MyBackupAgent extends BackupAgentHelper {
```

```

// The name of the SharedPreferences file
static final String PREFS = "myprefs";
// A key to uniquely identify the set of backup data
static final String PREFS_BACKUP_KEY = "myprefs";
@Override
public void onCreate() {
    SharedPreferencesBackupHelper helper = new SharedPreferencesBackupHelper(
        this, PREFS);
    addHelper(PREFS_BACKUP_KEY, helper);
}
}

```

## 6-12-4- راه اندازی و فعال کردن پروسه های restore و backup

### (بازگردانی داده ها)

پیاده Backup manager سازی شده، به صورت اتوماتیک و بدون دخالت و تعامل کاربر shared preferences و تنظیمات دلخواه ثبت شده کاربر را به طور دائمی ذخیره نموده و بازگردانی می نماید. کافی است shared preferences را تحت فایلی که اپلیکیشن خود را با نام آن ثبت کردید، تغییر داده و ذخیره نمایید.

adb shell command چنانچه مایلید عملیات ذخیره و بازگردانی داده ها را از طریق پنجره فرمان تست نمایید، در آن صورت می توانید دستورات زیر را به کار ببرید.

```

# Trigger a backup, usage:
# adb shell bmgr backup <package>
# schedule backup
adb shell bmgr backup com.vogella.android.databackup
# ensure scheduled backup run
adb shell bmgr run
# to restore you backup use bmgr restore
adb shell bmgr restore com.vogella.android.databackup

```

## بخش ششم :

### استفاده ای بهینه از حافظه و افزایش کارایی در اندروید

آموزش حاضر به شرح نحوه ای استفاده ای بهینه از حافظه و افزایش کارایی اپلیکیشن می پردازد. کدهای این بخش داخل محیط برنامه نویسی Android Studio تست و اجرا شده است.

#### 13-6-رهنمودهایی جهت طراحی و ارائه ای اپلیکیشن های کارامد

##### 13-6-چرا باید در استفاده از منابع اندروید مراقب بود؟

دستگاه های اندروید نسبت به کامپیوترهای رومیزی یا لپ تاپ قدرت و توانایی کمتری دارند. به همین دلیل بایستی در استفاده از حافظه دقت بیشتری داشته باشید. به ویژه از ویرایش 5.0 اندروید قبل تر، توسعه دهنده باید مراقب باشد تا حد امکان garbage collector دستگاه مجازی جاوا (JVM) را درگیر نکند چرا که این امر سبب freeze و هنگ کردن runtime اندروید به مدت 200 میلی ثانیه می شود. در چنین شرایطی چنانچه کاربر، برای مثال، با نوار اسکرول در حال پیمایش به پایین لیست باشد، با توجه به تأخیر قابل توجهی که garbage collector سبب می شود، تجربه ای کاربری به شدت پایین می آید.

##### 13-6-اجتناب از تخصیص و ایجاد آبجکت غیر ضروری

سعی کنید تا حد امکان از ایجاد آبجکت های غیر ضروری، به خصوص در جاهای هزینه بر خودداری نمایید. توصیه می کنیم آبجکت های موجود را بازیافت نموده و مجددا از آن ها استفاده کنید. ایجاد آبجکت های اضافی و غیر ضروری سبب راه اندازی مکرر garbage collector می شود. می توان با اجتناب از ساخت آبجکت های غیر ضروری به راحتی از این رخداد جلوگیری نمود.

برای مثال بهتر است از ایجاد آبجکت در حلقه ها یا در بدنه ی متدهای `onDraw()` داخل آبجکت `view` اختصاصی خود خودداری نمایید.

### 6-13-3-استفاده از `data structure` های کارآمد

اندروید چندین نمونه با پیاده سازی های مختلف از کلاس های `Sparse*Array` ارائه می دهد. به کد زیر توجه کنید.

```
Map<Integer, String> map = new HashMap<Integer, String>();
```

استفاده از این کد سبب ساخت آبجکت های غیر ضروری از نوع `Integer` می شود.

محیط اندروید جهت نگاشت مقادیر به دیگر آبجکت ها، `data structure` های کارمتری را در اختیار توسعه دهنده قرار می دهد. در صورت امکان توصیه‌ی ما بر این است که از این ساختارها استفاده نمایید چرا که این ساختارها از ساخت آبجکت اضافی جلوگیری می کنند. از جمله می توان به `HashMap` اشاره کرد.

همان طور که در بالا نیز گفته شد، ساخت آبجکت عملیات سنگین و هزینه بری بوده و تا حد امکان می بایست از رخداد غیر ضروری آن جلوگیری نمود. زیرا به دنبال کاهش تعداد دفعات ایجاد آبجکت، `garbage collector` نیز دفعات کمتری جهت مدیریت حافظه فراخوانده می شود.

جدول زیر مثال هایی از کاربرد `SparseArrays` را نمایش می دهد.

Data structure هایی که از نظر مصرف حافظه بهینه هستند	
Data structure	شرح
<code>SparseArray&lt;E&gt;</code>	مقادیری از نوع عدد صحیح را به آبجکت ها نگاشت کرده و از ساخت آبجکت های <code>integer</code> جلوگیری می کند.
<code>SparseBooleanArray</code>	مقادیر از نوع <code>integer</code> را به بولی ها ( <code>boolean</code> ) نگاشت می کند.

SparseIntArray

مقادیر integer را به integer ها نگاشت می کند.

به منظور بهینه سازی مثال کاربردی فوق، بهتر است از **data structure** زیر استفاده نمایید.

```
SparseArray<String> map = new SparseArray<String>();
map.put(1, "Hello");
```

## 14-6- مدیریت bitmap

Bitmap ها (عکس های پیکسلی) ، چنانچه در اندازه ی کامل و حقیقی بارگذاری شوند، مقدار زیادی از حافظه را اشغال خواهند کرد. توصیه می شود bitmap ها را فقط در اندازه ی مورد نیاز بارگذاری کنید.

فرض کنید اپلیکیشنی طراحی کردید که تصویری در ابعاد dp 100x100 را نمایش می دهد، برای این اپلیکیشن می بایست تصویر مربوطه را دقیقا در همین اندازه داخل حافظه بارگذاری نمایید.

یک روش معمول این است که با ارسال یک flag به **BitmapFactory**، **BitmapFactory.Options** دلخواه را بدون اینکه در حافظه بارگذاری شود، اول اندازه گیری نمایید.

```
// instruct BitmapFactory to only the bounds and type of the image
BitmapFactory.Options options = new BitmapFactory.Options();
options.inJustDecodeBounds = true;
BitmapFactory.decodeResource(getResources(), R.id.myimage, options);
// get width and height
int imageHeight = options.outHeight;
int imageWidth = options.outWidth;
// type of the image
String imageType = options.outMimeType;
```

پس از آن می توانید نسخه‌ی درست اندازه بندی شده‌ی تصویر مورد نظر را در حافظه بارگذاری نمایید. اندروید تصاویر را به توان 2 می‌رساند. می‌توانید با استفاده از متاد زیر (ضریب مقیاس بندی) را بر مبنای 2 محاسبه نمایید.

```
public static Bitmap decodeBitmapWithGiveSizeFromResource(Resources res, int resId,
    int reqWidth, int reqHeight) {
    // First decode with inJustDecodeBounds=true to check dimensions
    final BitmapFactory.Options options = new BitmapFactory.Options();
    options.inJustDecodeBounds = true;
    BitmapFactory.decodeResource(res, resId, options);
    // Calculate inSampleSize
    options.inSampleSize = calculateInSampleSize(options, reqWidth, reqHeight);
    // Decode bitmap with inSampleSize set
    options.inJustDecodeBounds = false;
    return BitmapFactory.decodeResource(res, resId, options);}
public static int calculateInSampleSize(
    BitmapFactory.Options options, int reqWidth, int reqHeight) {
    // Raw height and width of image
    final int height = options.outHeight;
    final int width = options.outWidth;
    int inSampleSize = 1;
    if (height > reqHeight || width > reqWidth) {
        final int halfHeight = height / 2;
        final int halfWidth = width / 2;
        // Calculate the largest inSampleSize value that is a power of 2 and keeps both
        // height and width larger than the requested height and width.
        while ((halfHeight / inSampleSize) > reqHeight
            && (halfWidth / inSampleSize) > reqWidth) {
            inSampleSize *= 2;}
    }
    return inSampleSize;}
```

با استفاده از این متاد می‌توان تصویر مورد نظر را همان طور که در مثال زیر نمایش داده شده، به view تخصیص داد.

```
viewWidth = imageView.getWidth();
viewHeight = imageView.getHeight();

imageView.
imageView.setImageBitmap(
    decodeSampledBitmapFromResource(getResources(), R.id.myimage, viewWidth, viewHeight));
```

## Cache-15-6-استفاده از Cache

### 15-6-استفاده از حافظه i نهان

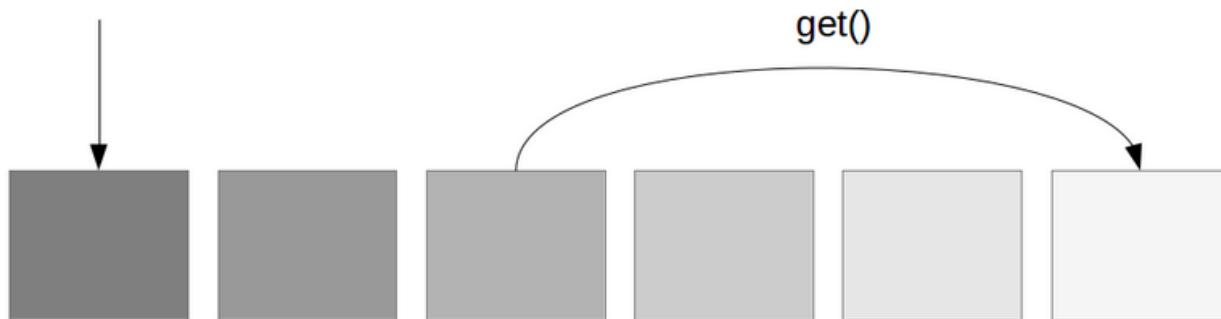
همان طور که می دانید فرایند ساخت آبجکت بسیار سنگین و هزینه بر است. Cache این امکان را فراهم می کند تا آبجکت هایی که قبلا در حافظه موجود هستند را بازیافت نموده (مجدداً) مورد استفاده قرار داد) و بدین وسیله از تخصیص یافتن مجدد منابع به ساخت آبجکت (جدید) خودداری گردد. به عبارتی، زمانی که شما آبجکتی را در حافظه بارگذاری می کنید، می توانید این حافظه را به عنوان cache و حافظه i موقت برای میزبانی آبجکت ساخته شده در نظر بگیرید. به عنوان مثال زمانی را در نظر بگیرید که تعدادی فایل تصویری را از اینترنت بارگیری می کنید. در این لحظه، جهت نمایش فایل های ذکر شده در یک لیست، می بایست آن ها را به طور موقت در حافظه نگه دارید. امر مانع از این می شود که عکس ها بارها از اینترنت دانلود شوند.

شما به ناگذیر بایستی برخی از آبجکت های مستقر در حافظه را بازیافت نمایید چرا که در غیر این صورت با کمبود حافظه مواجه خواهید شد. یک روش پیشنهادی این است که آبجکت هایی که طولانی مدت در اپلیکیشن مورد استفاده قرار نگرفته اند را بازیافت نمایید.

محیط (platform) اندروید از API 12 (ورژن 12 کتابخانه های اندروید) کلاسی به نام LruCache را ارائه می نماید (لازم به ذکر است که این قابلیت در کتابخانه های پشتیبانی از ورژن های قبلی اندورید support-v4 library نیز قابل دسترسی می باشد). کلاس مزبور امکان پیاده سازی یک cache جهت نگهداری آبجکت هایی که مدت طولانی مورد استفاده قرار نگرفته اند را فراهم می سازد. به عبارت دیگر این کلاس یک LRU cache پیاده سازی می کند. در این cache تمامی اعضایی که مدت ها از آخرین بار استفاده از آن ها می گذرد، قرار می گیرند (آبجکت هایی که کمترین استفاده i اخیر را داشته اند در این حافظه i موقت جای می گیرند). این cache ظرفیت مشخصی دارد و هنگامی که این ظرفیت پر می شود، تمامی آیتم هایی که طولانی مدت مورد استفاده قرار نگرفته اند بر حسب نیاز از این cache حذف می شوند. این قابلیت تصویر زیر نمایش داده شده است.

# LRU Cache

Least used  
items in the list



Calling get() for an item, moves it to the top of the cache



مثال زیر یک نمونه از پیاده سازی کلاس LruCache را برای ذخیره ی موقتی فایل های تصویری فراهم می سازد.

```
public class ImageCache extends LruCache<String, Bitmap> {
    public ImageCache( int maxSize ) {
        super( maxSize );
    }
    @Override
    protected int sizeOf( String key, Bitmap value ) {
        return value.getByteCount();
    }
    @Override
    protected void entryRemoved( boolean evicted, String key, Bitmap oldValue, Bitmap newValue ) {
        oldValue.recycle();
    }
}
```

همان طور که در نمونه کد زیر مشاهده می کنید، استفاده از آن بسیار ساده می باشد.

```
LruCache<String, Bitmap> bitmapCache = new LruCache<String, Bitmap>();
```

جهت اطلاع از ظرفیت حقیقی cache، کافی است اندازه و ظرفیت کل حافظه را قابل استفاده در دستگاه میزان را بررسی نمایید. برای این منظور می توانید از کلاس MemoryClass استفاده نمایید. کد زیر استفاده از آن را به نمایش می گذارد.

```
int memClass = ( ( ActivityManager) activity.getSystemService( Context.ACTIVITY_SERVICE )  
).getMemoryClass();  
int cacheSize = 1024 * 1024 * memClass / 8;  
LruCache cache = new LruCache<String, Bitmap>( cacheSize );
```

## 6-15-2- پاک سازی cache

از API 14 این امکان برای شما وجود دارد که متده onTrimMemory() را در کامپوننت های نرم افزاری اندورید بازنویسی (override) نمایید. این متده را اندروید فراخوانی نموده و به شما این اجازه را می دهد تا زمانی که سیستم منابعی را جهت پردازش های پیش زمینه ای لازم داشت، محتوای حافظه را پاک سازی و آزاد نمایید.

آموزشگاه تحلیلگر داده



621

آدرس آموزشگاه : تهران - خیابان شریعتی - بالاتر از خیابان ملک - جنب بانک صادرات - پلاک 651 طبقه دوم - واحد 7

88146323 - 88446780 - 88146330

## بخش اول :

# تست بخش های مجزای اپلیکیشن های اندرویدی به وسیلهٔ ابزار unit testing / آزمایش کامپوننت های اپلیکیشن اندروید با استفاده از ابزار UUnit

آموزش حاضر به شرح unit testing (اجرای تست نرم افزاری بر روی بخش های مختلف پروژه به صورت مجزا) با استفاده از ابزار xUnit می‌پردازد. این آموزش نحوهٔ ایجاد تست های Unit را شرح داده و سپس استفاده از محیط برنامه نویسی Eclipse را در راستای نوشتتن تست های نرم افزاری لازم جهت کسب اطمینان از عملکرد صحیح بخش های مختلف اپلیکیشن را توضیح می‌دهد.

### 1-7-هدف از نوشتتن تست های نرم افزاری چیست؟

#### 1-7-1-شرح مفهوم تست های نرم افزاری

تست اپلیکیشن (software test) یک قطعهٔ نرم افزاری (کد) هست که خود یک قطعهٔ نرم افزاری دیگر را (جهت آزمایش و اطمینان از کارکرد صحیح و اینکه آیا قطعهٔ نوشته شده هدف مورد نظر را برآورده می‌سازد) اجرا می‌کند. این تست نرم افزاری بررسی می‌کند آیا قطعهٔ کد نوشته شده منجر به وضعیت مورد انتظار می‌شود (که به آن state testing گویند) و یا مراحل لازم (سلسهٔ رخداد ها) به ترتیب رخ می‌دهند یا خیر (که به آن behavior testing گویند).

## 7-1-2-چرا استفاده از تست ها توصیه می شود؟

Unit test به توسعه دهنده کمک می کند تا منطق یک تکه از برنامه را آزمایش کرده و بدین وسیله مطمئن شود رفتار مورد انتظار را از خود نشان می دهد/هدف و نیاز مدنظر را برآورده می سازد یا خیر.

در واقع با نوشتمن و اجرای unit test بر روی بخش های نرم افزاری به صورت مجزا این امکان فراهم می شود تا کاستی ها و خطاهایی که ممکن است با اضافه شدن رفتار / اعمال تغییرات جدید به برنامه در آن رخنه کرده باشند، توسط توسعه دهنده به سرعت کشف شده و برطرف گردد. نوشتمن تست برای بخش های متعددی از اپلیکیشن به برنامه نویس کمک می کند تا بعده ها بدون اینکه مجبور به نوشتمن تست های دستی (manual test) شود، بتواند قابلیت های نوین زیادی با عملکرد صحیح به اپلیکیشن خود اضافه نماید.

## 7-1-3-فریم ورک های تست گیری (testing frameworks) برای Java

فریم ورک های تست گیری متعددی برای Java نوشته شده و در دسترس می باشد. از جمله‌ی پرکاربرد و محبوب ترین آن‌ها می‌توان به JUnit و TestNG اشاره کرد.

مقاله‌ی حاضر بر روی نحوه استفاده از ابزار JUnit مرکز می شود.

## 7-2-واژه ها و مفاهیم مرتبط با تست گیری

### 7-2-1-کد (یا اپلیکیشن) مورد تست / Code Under Test

کدی که تست بر روی آن اجرا می شود در اصطلاح Code Under Test (کد مورد آزمایش) خوانده می شود. حال اگر واحد مورد تست شما یک آپلیکیشن باشد، در آن صورت application under test اطلاق خواهد شد.

## (مقادیر ثابت تست) Test fixture-7-2-2

عبارت است از state ثابت (اطلاعات مربوط به وضعیت) چند آبجکت که به عنوان مقادیر پایه (baseline) برای اجرای تست بر روی بخش های مجزا نرم افزار مورد استفاده قرار می گیرد. مقصود اصلی استفاده از test fixture این است که محیط ثابت و مشخصی برای تست وجود داشته باشد به طوری که نتایج / خروجی حاصل تکرارپذیر باشد. می توان از آن به عنوان پیش شرط های تست نیز نام برد. به عبارت دیگر، هرچه که بایستی از قبل آماده باشد تا یک تست اجرا شده و نتیجه ی مورد نظر را به عنوان خروجی انتظار داشت، text fixture خوانده می شود.

به عنوان مثال، می تواند یک رشته ی ثابت باشد که به عنوان پارامتر ورودی ثابت به یک متده می شود. در نهایت تست مشخص می کند آیا متده مورد آزمایش نتایج مورد انتظار را تولید می کند / عملکرد صحیح را ارائه می دهد یا خیر.

## 7-2-3-شرح مفهوم Unit test و تست بخش های مختلف نرم افزار (کد) به صورت مجزا

یک قطعه کد است که توسعه دهنده برای تست قابلیتی (کسب اطمینان از عملکرد صحیح کد) از برنامه آن را نوشته و انتظار دارد که نتیجه ی (رفتار یا وضعیت) دلخواه را تولید کند.

آن درصدی از کل کد که مورد آزمایش قرار می گیرد، در اصطلاح test coverage اطلاق می گردد.

Unit test معمولاً بخش یا قطعات (مستقل) کوچکی از برنامه را در سطح کلاس یا متده آزمایش قرار می دهد. لازم به ذکر است که dependency ها و کتابخانه های خارجی را می بایست از unit test ها حذف کرد. این کار را می توان با جایگزین کردن این کتابخانه ها با پیاده سازی و بدنه ی تست یا آبجکت ساختگی که از پیش توسط framework ساخته شده، انجام داد.

گفتنی است که Unit test ها کارایی چندانی برای تست رابط های کاربری پیچیده و تعامل داده بین کامپوننت های نرم افزاری ندارند. برای این منظور بهتر است integration test طراحی نمایید.

## Integration test-7-2-4 ( تست اپلیکیشن در context و بستر خودش )

عبارت است از آزمایش اپلیکیشن در context و بستر کلی خودش به طوری که برای تست نیازمند بخش های دیگر اپلیکیشن باشد (به عبارت دیگر آن قطعه هایی از کد که بر روی بخش های دیگر تاثیر جانبی side effect دارند). Integration test مشترک چند کامپوننت نرم افزاری تمرکز می کند. این نوع تست همچنین اطمینان حاصل می کند که کل سیستم به درستی و مورد انتظار رفتار می کند و در این حین نیاز برای نوشتن و اجرای تست های دقیق دستی را کاهش می دهد.

این نوع تست ها به شما امکان می دهد تا پیشنهادات و بازخورد کاربران از اپلیکیشن را در مجموعه تست هایی (test suite) که برای آزمایش و کسب اطمینان از عملکرد صحیح اپلیکیشن نوشته اید، لاحظ نمایید. از این جهت، خروجی تست مربوطه مشابه تعامل مورد انتظار بین کاربر و اپلیکیشن خواهد بود.

## Performance test-7-2-5 ( تست های بررسی کارایی )

Performance test یا تست های ارزیابی کارایی به طور مکرر برای بررسی میزان کارایی و مستند سازی قدرت کامپوننت های نرم افزاری مورد استفاده قرار می گیرند. تست ذکر شده را توسعه دهنده برای این می نویسد که میزان کارایی کد مورد آزمایش را در شرایطی که بار کاری سنگین هست بررسی نموده و از بالا بودن بازده در هر شرایطی اطمینان حاصل کند.

## State testing و Behavior testing-7-2-6

تستی که بررسی می کند آیا متدهایی با پارامترهای ورودی مورد نظر فراخوانی شده اند یا خیر، خوانده می شوند. Behavior test (behavior test) نتیجه ی متدهای فراخوانی شده را اعتبار سنجی نمی کند.

( تست وضعیت ) ، برخلاف behavior test ، در بررسی و آزمایش رفتار اپلیکیشن مورد تست خلاصه می شود.

چنانچه الگوریتم یا قابلیت های سیستمی را مورد آزمایش قرار داده اید ، در آن صورت تست شما بایستی بر روی state متمرکز شود ، نه تعامل بین کامپوننت ها .

در این تست ها غالبا از آبجکت های ساختگی (mock) و stub (کد جایگزین) کلاس های مرتبط برای استخراج ، حذف تعامل یا وابستگی با دیگر کلاس ها استفاده می شود . پس از آن شما می توانید (اطلاعات مربوط به) وضعیت یا رفتار را بر اساس نیاز خود تست نمایید .

### 3-3-سازماندهی تست

#### 3-3-1- تست در کجا بایستی نوشته شود

Unit test های معمولی اغلب در پروژه ی مجزا یا پوشه ی اصلی (source folder) جداگانه می شوند . این کار سبب می شود کد تست شده از کد اصلی جدا نگه داشته باشد .



#### 3-3-2- کدام بخش نرم افزار بایستی تست شود

اینکه دقیقا می بایست چه بخش هایی تست شود ، یک موضوع بسیار بحث برانگیز است . برخی از توسعه دهندگان بر این عقیده هستند که تمامی دستورات داخل کد می بایست مورد بررسی قرار گرفته و تست شود .

به هر حال شما بایستی برای بخش های پیچیده و حساس نرم افزار خود تست بنویسید . اگر ویژگی و قابلیت های جدید به اپلیکیشن خود اضافه نمایید ، آنگاه یک مجموعه تست قدرتمند و کارامد می تواند اپلیکیشن شما را در برابر ضعف و خطاهایی که ممکن است بعدها به داخل اپلیکیشن و کد جاری راه پیدا کنند محافظت خواهد کرد .

در طراحی تست نرم افزاری می توان بخش های جزئی از کد را نادیده گرفت. برای مثال، نوشتن تست برای متدهای getter و setter که صرفا مقادیری را به فیلدهایی تخصیص می دهند، تقریبا بیهوده است. در واقع نوشتن تست برای تمامی دستورات داخل کد مشخصا بسیار زمان بر بوده و تقریبا امری بی فایده می باشد چرا که شما اپلیکیشن را در بستر JVM تست می کنید. JVM به خودی خود test case هایی از پیش تعریف شده و مشخص دارد.

در صورت طراحی تست برای code base هایی که قبلاً تستی برای آن ها نوشته نشده، بهتر است ابتدا برای آن بخش هایی از کد تست تهیه طراحی نمایید که دفعات رخداد خطأ در گذشته در آن ها زیاد باشد. سپس می توانید بر روی بخش های اساسی کد (اپلیکیشن) متمرکز شوید.



## 4-7 استفاده از JUnit

### 7-4-1 JUnit framework (فریم ورک انجام unit test بر روی پروژه های جاوا)

JUnit 4.x در واقع یک فریم ورک اجرای unit test بر روی پروژه های جاوای است که با استفاده از annotation ها، متدهایی که رفتارها و قابلیت های تست را به صورت جداگانه آزمایش می کنند با علائم یا دستورهای خاص نشانه گذاری و معرفی می نماید. برای کسب اطلاعات بیشتر در زمینه ای این فریم ورک تست گیری می توانید به آدرس <http://junit.org> و <https://github.com/junit-team/junit> مراجعه نمایید.

## ۷-۴-۲-نحوه‌ی طراحی یک تست در فریم‌ورک تست گیری JUnit؟

JUnit test در حقیقت یک متاد است که داخل کلاس تعریف شده و منحصرا برای منظور تست مورد استفاده قرار می‌گیرد. کلاسی که میزبان متاد مذکور می‌باشد در اصطلاح `test class` نامیده می‌شود. به منظور نوشتن یک تست بر مبنای ۴ JUnit، متاد مورد نظر را با دستور `@org.junit.Test` داخل کلاس میزبان نشانه گذاری (annotate) می‌کنید.

این متاد کد مورد آزمایش را اجرا می‌کند. می‌توانید از متاد `assert` که خود JUnit ارائه می‌دهد استفاده نمایید و یا از متادی که فریم‌ورک assert دیگری فراهم می‌نماید، بهره بگیرید. متاد `assert` در اصل برای بررسی نتیجه‌ی واقعی با نتایج مورد انتظار مورد استفاده قرار می‌گیرد. متدهایی که به این صورت استفاده می‌شوند در اصطلاح تست گیری، `assert`‌ها یا دستورات `assert` خوانده می‌شوند.

لازم است در دستورات `assert` پیغام‌های معنی دار قرار دهید چرا که در صورت مواجه شدن با خطأ کاربر سریع‌تر می‌تواند مشکل را شناسایی کرده و برطرف نماید. این ویژگی می‌تواند به ویژه برای فردی که کد را می‌خواند اما در نوشتن آن شرکت نداشته، مفید باشد.

## ۷-۴-۳-نمونه‌ای از JUnit test

کد زیر یک نمونه از پیاده سازی تست JUnit را نمایش می‌دهد. این تست فرض را بر این می‌گذارد که کلاس MyClass وجود داشته و متادی به نام `multiply(int, init)` را دربردارد.

```
import static org.junit.Assert.assertEquals;
import org.junit.Test;
public class MyTests {
    @Test
    public void multiplicationOfZeroIntegersShouldReturnZero() {
        MyClass tester = new MyClass(); // MyClass is tested
        // assert statements
        assertEquals("10 x 0 must be 0", 0, tester.multiply(10, 0));
        assertEquals("0 x 10 must be 0", 0, tester.multiply(0, 10));
        assertEquals("0 x 0 must be 0", 0, tester.multiply(0, 0));
    }
}
```

## 4-4-7- قرارداد/روش های نام گذاری تست های مبتنی بر JUnit

قراردادهای نام گذاری متعددی برای تخصیص اسم به تست های طراحی شده بر اساس فریم ورک JUnit وجود دارد. یک راه حل پر طرفدار افزودن پسوند "-test" به اسم کلاس های تست گیری (test case) و جایگذاری آن ها در پکیج جدید "test" می باشد.

به طور کلی، اسم تست باید دربارهٔ کاربرد و مورد استفادهٔ آن توضیح دهد. در صورتی که اسم تخصیص داده شده به تست، مورد کاربرد آن را روشن بیان نماید، دیگر نیازی به خواندن پیاده سازی و محتوای کد نخواهد بود.

یک قرارداد ممکن و پرطرفدار استفاده از کلمهٔ "should" در اسم متد تست گیری می باشد. برای مثال، می توان به "menuShouldGetActive" یا "ordersShouldBeCreated" اشاره کرد. با این نوع انتخاب اسم برای متد، مشخص می شود که متد مورد نظر دقیقاً چه رفتاری را تست کرده و انتظار چه نتیجه‌ای را دارد.

## 4-5-7- قراردادهای نام گذاری JUnit برای Maven

در صورت استفاده از سیستم کامپایل (build system) Maven، بهتر است از "Test" بجای "Tests" به عنوان پسوند استفاده نمایید. سیستم کامپایل Maven (به واسطهٔ افزونهٔ surefire) به صورت اتوماتیک و درون ساخته این کلاس ها را داخل test scope خود دربردارد.

## 4-6-7- مجموعه تست های JUnit (JUnit test suites)

در صورتی که چندین کلاس تست گیری برای اپلیکیشن نوشته باشید، بد نیست تمامی آن ها را در یک قالب واحد به نام test suite یا مجموعه تست بگنجانید. شایان ذکر است که اجرای test suite به مثابهٔ اجرای تمامی کلاس های تست گیری، به ترتیب از قبل مشخص شده است که در مجموعه تست تعریف شده اند.

یک مجموعهٔ بزرگ از تست ها است که خود می تواند test suite های دیگر را دربرگیرد.

نمونه کد زیر نحوه‌ی استفاده از test suite را نمایش می‌دهد. حاضر دو کلاس تست گیری MySecondClassTest و MyClassTest را شامل می‌شود. در صورت تمایل می‌توانید به این کلاس‌های تست گیری، کلاس دیگری اضافه نمایید. کافی است کلاس مورد نظر را به دستور `@Suite.SuiteClasses`

```
package com.vogella.junit.first;
import org.junit.runner.RunWith;
import org.junit.runners.Suite;
import org.junit.runners.Suite.SuiteClasses;
@RunWith(Suite.class)
@SuiteClasses({
    MyClassTest.class,
    MySecondClassTest.class })
public class AllTests { }
```

#### 7-4-7- اجرای تست از طریق خط فرمان/ command line

می‌توانید تست مورد نظر را خارج از محیط برنامه نویسی (IDE) و به وسیله‌ی کدهای استاندارد Java راه اندازی کنید. برنامه نویس قادر خواهد بود با ترکیب سیستم‌های کامپایلی (build system) (Jenkins) نظیر Apache Maven یا Gradle با یک Continuous Integration Server (همچون Jenkins) تست‌ها را به صورت خودکار و در فواصل زمانی منظم بر روی بخش‌هایی از نرم افزار به اجرا در آورد.

کلاس `org.junit.runner.JUnitCore` متدهی به نام `runClasses()` را ارائه می‌دهد. این متدهی توسعه دهنده امکان می‌دهد تا همزمان یک یا چند کلاس تست گیری (test class) را اجرا کند. کلاس مذبور در خروجی آبجکتی از جنس `org.junit.runner.Result` برمی‌گرداند. این آبجکت را می‌توان جهت بازیابی اطلاعات مربوط به تست‌ها مورد استفاده قرار داد.

کلاس زیر نمایش می‌دهد چگونه می‌توان `MyClassTest` را اجرا نمود. این کلاس، کلاس تست گیری (test class) شما را اجرا کرده و خطاهای ممکن را در `console` درج می‌نماید.

```

package de.vogella.junit.first;
import org.junit.runner.JUnitCore;
import org.junit.runner.Result;
import org.junit.runner.notification.Failure;
public class MyTestRunner {
    public static void main(String[] args) {
        Result result = JUnitCore.runClasses(MyClassTest.class);
        for (Failure failure : result.getFailures()) {
            System.out.println(failure.toString());
        }
    }
}

```

کلاس حاضر را می توان مانند سایر برنامه های java در خط فرمان اجرا نمود. کافی است فایل JAR کتابخانه ی JUnit را به classpath اضافه نمایید.

## 7-5-ساختمانهای پایه ای فریم ورک JUnit

### JUnit های Annotation-7-5-1

JUnit با استفاده از annotation های خود، متدهای تست گیری را علامت گذاری کرده و امکان تنظیم اختصاصی آن ها را برای توسعه دهنده فراهم می آورد. جدول زیر لیستی از های مهم در فریم ورک JUnit را به نمایش می گذارد.

JUnit Annotations	
Annotation	شرح
@Test public void method()	حاضر نشانگر این است که متد مورد نظر، یک متد تست گیری / آزمایشی است.
@Test (expected = Exception.class)	در صورتی که متد حاضر، exception نام بده را صادر نکرد، با شکست مواجه می شود.
@Test(timeout=100)	در صورتی که اجرای متد بیش از 100 میلی ثانیه به طول انجامید، با شکست مواجه می شود.
@Before public void method()	این متد قبل از هر تست اجرا می شود. در واقع با استفاده از این متد محیط برای تست گیری آماده می شود(عملیات ابتدایی همچون خواندن داده های ورودی، مقداردهی اولیه کلاس انجام می پذیرد).

<pre>@After public void method()</pre>	بعد از هر تست اجرا می شود. از این متده برای پاک سازی محیط تست گیری (همچون حذف داده های موقتی، بازگردانی تنظیمات پیش فرض) استفاده می شود. بعلاوه قادر است با حذف ساختارهای هزینه بر و سنگین در حافظه، میزان استفاده از آن را کاهش دهد.
<pre>@BeforeClass public static void method()</pre>	این متده یکبار قبل از اجرای تمامی تست ها اجرا می شود. متده مذکور برای اجرای عملیات زمان بر، برای مثال، اتصال به دیتابیس مورد استفاده قرار می گیرد. توابعی که با این annotation نشانه گذاری شده باشند، می بایست برای کار (سازگاری) با JUnit به صورت static تعریف شده باشند.
<pre>@AfterClass public static void method()</pre>	این متده یکبار، پس از اینکه تمام تست ها انجام شده و به پایان رسیدند، اجرا می شود. با فراخوانی این متده، عملیات پاک سازی همچون قطع اتصال به دیتابیس صورت می گیرد. متدهایی که با این annotation نشانه گذاری شده اند، می بایست جهت سازگاری با JUnit به صورت static (با کلیدواژه static) تعریف شده باشند.
<pre>@Ignore or @Ignore("Why disabled")</pre>	با درج این annotation بالای متده، متده تست گیری نادیده گرفته شده یا غیرفعال می شود. این به خصوص برای زمانی که کدهای اصلی تغییر یافته ولی test case هنوز متناسب با آن ویرایش و تنظیم نشده، یا اگر مدت اجرای تست بیش از حد طولانی شده باشد، مفید واقع می شود. توصیه می شود پارامتر اختیاری را نیز لحاظ نموده و مقداردهی نمایید (چرا تست غیرفعال شده است).

## 7-5-2-Assert دستورات

JUnit کلاسی به نام Assert`class را شامل می شود. این کلاس توابع static ای را ارائه می دهد که شرایط خاصی را چک می کنند. دستورات assert معمولاً با `assert آغاز می شوند. متده assertion، خروجی واقعی تست را با مقدار مورد انتظار مقایسه می کند و چنانچه مقایسه با شکست مواجه شد، یک AssertionException صادر می کند.

جدول زیر این توابع را به صورت اجمالی همراه با شرح عملکرد هریک به نمایش می گذارد.

assert	توابع
--------	-------

پارامترهای ذکر شده در [], اختیاری بوده و از جنس String هستند.

assert	توابع
--------	-------

دستور	شرح
fail(message)	تست با شکست مواجه شده و در خروجی پیغامی اختیاری نمایش داده می شود. توسعه دهنده ممکن است از این annotation برای غیرفعال کردن بخشی از کد یا مانع از جرا شدن آن، استفاده کند.
assertTrue([message,] boolean condition)	بررسی می کند آیا شرط بولی صحیح است یا خیر.
assertFalse([message,] boolean condition)	بررسی می کند آیا نتیجه ی شرط بولی غلط است یا خیر.
assertEquals([message,] expected, actual)	تست می کند آیا دو مقدار با هم برابر هستند یا خیر. توجه: درباره ی آرایه باید گفت که صرفا reference آن بررسی می شود و محتواهای آرایه ها مورد توجه قرار نمی گیرد.
assertEquals([message,] expected, actual, tolerance)	تست می کند آیا مقادیر float یا double با هم مطابقت دارند یا خیر. پارامتر tolerance تعداد ارقام اعشاری است که باید یکسان باشند.
assertNull([message,] object)	بررسی می کند آیا آبجکت مورد تست null هست یا خیر.
assertNotNull([message,] object)	بررسی کرده و اطمینان حاصل می نماید که آبجکت null نباشد.
assertSame([message,] expected, actual)	اطمینان حاصل می کند که هر دو متغیر به یک آبجکت اشاره داشته باشند.
assertNotSame([message,] expected, actual)	بررسی می کند که دو متغیر، هر یک به آبجکت جداگانه اشاره داشته باشند.

### 3-5-7-ترتیب اجرای تست

فریم ورک اجرای تست JUnit فرض را بر این می گذارد که تمامی متدهای آزمایشی (test method) با ترتیبی غیر مشخص و دلخواه اجرا شوند. در واقع تستی که به صورت کارامد نوشته شده باشد، نباید ترتیب اجرای خاصی داشته باشد. تست هایی که برای آزمایش بخش های مختلف برنامه نوشته می شود، نباید به سایر تست ها وابستگی داشته باشد.

از 4.11 JUnit به بعد، به صورت پیش فرض توصیه می شود از ترتیب deterministic و نه predictable برای اجرای تست بر روی نرم افزار استفاده نمایید.

می توانید با استفاده از یک annotation مشخص کنید که متدهای تست گیری (test method) بر اساس اسم متدهای شوند (در واقع بر اساس ترتیب حروف الفبا). به منظور فعال سازی این ویژگی، بایستی کلاس تست گیری (test class) خود را با `@FixMethodOrder(MethodSorters.NAME_ASCENDING)` (annotate) علامت گذاری نمایید. همچنین می توانید با استفاده از پارامتر `annotation` در `MethodSorters.DEFAULT`، به صورت صریح، حالت پیش فرض را تنظیم نمایید. بعلاوه می توانید از `MethodSorters.JVM` که از تنظیمات پیش فرض JVM استفاده می کند، بهره بگیرید که ممکن است از یک اجرا به اجرای دیگر متفاوت باشد.

#### 4-5-7-غیرفعال سازی تست ها

@Ignore این امکان را فراهم می آورد که یک تست را به صورت static غیرفعال کرده و در اصطلاح آن را در تست نادیده بگیرید. یا می توانید با استفاده از پارامترهای `Assume.assumeTrue` یا `Assume.assumeFalse` یک شرط برای تست تعریف نمایید. در صورتی که شرط نتیجه `true` بدهد و برقرار باشد، پارامتر `Assume.assumeFalse` تست مربوطه را نامعتبر (invalid) معرفی و نشانه گذاری می کند. چنانچه شرط نتیجه `false` بدهد، پارامتر `Assume.assumeTrue` پس از ارزیابی تست، آن را نامعتبر معرفی و علامت گذاری می کند. به طور مثال، دستور زیر یک تست را در سیستم عامل Linux را غیرفعال می کند:

```
Assume.assumeFalse(System.getProperty("os.name").contains("Linux"));
```

#### 7-6-پشتیبانی محیط برنامه نویسی از JUnit

##### 7-6-1-ایجاد تست های JUnit

در صورت تمایل می توانید تست های JUnit را خود به صورت دستی بنویسید، اما محیط کاری Eclipse این امکان را برای شما فراهم می آورد تا تست های JUnit را به صورت ویژادی و طبق یک روال از پیش تعیین شده تهیه نمایید.

برای مثال، می توان از ایجاد تست JUnit یا یک کلاس تست گیری جهت آزمایش عملکرد کلاس از پیش نوشته شده نام برد. برای این منظور بر روی کلاس جدید راست کلیک کرده، کلاس مورد نظر را از Package Explorer view انتخاب نمایید. سپس بار دیگر بر روی آن راست کلیک کرده و مسیر New ▶ JUnit Test Case را طی کنید.

در صورت تمایل می توانید این کار را به صورت ویزاردی و با استفاده از JUnits Wizards انجام دهید. برای دستیابی به این راهنمای نصب ویزاردی، کافی است مسیر رو به رو را دنبال کنید: .File ▶ New ▶ Other... ▶ Java ▶ JUnit

## 7-6-2-اجرای تست های JUnit

محیط کاری Eclipse به توسعه دهنده این امکان را می دهد تا تست ها را به صورت تعاملی اجرا کند.

به منظور اجرای یک تست، کلاس تست گیری (test class) را انتخاب نموده، سپس بر روی آن راست کلیک کنید. حال گزینه‌ی Run-as ▶ JUnit Test را انتخاب کنید.

این کار JUnit را راه اندازی کرده و تمامی متدهای تست گیری (test method) را داخل کلاس اجرا می کند.

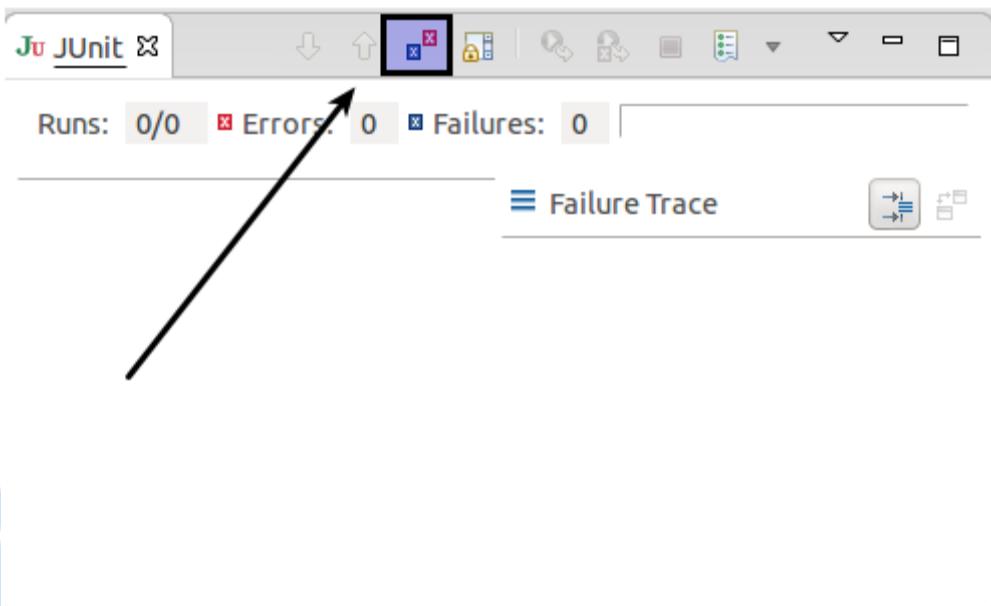
به برنامه نویس اجازه می دهد تا با فشردن کلیدهای Alt+Shift+X به صورت همزمان، تست مورد نظر را در کلاس انتخابی اجرا کند.

برای اینکه فقط و فقط تست انتخابی اجرا شود، اشاره گر موس را بر روی اسم متدهای تست گیری قرار داده و سپس کلیدهای فوق را فشار دهید.

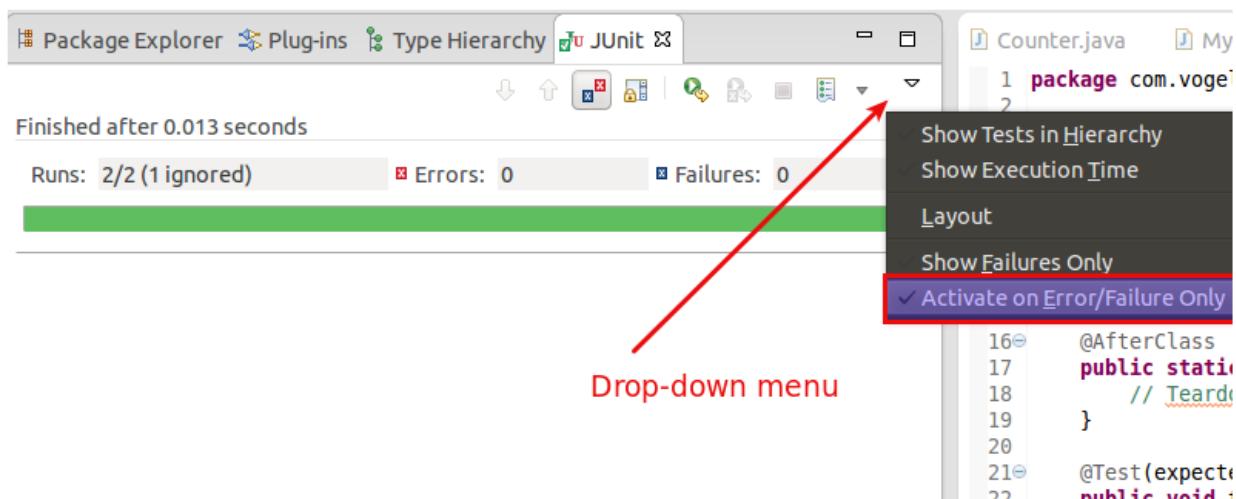
به منظور مشاهده نتیجه‌ی تست JUnit، محیط کاری Eclipse یک ابزار به نام view JUnit را ارائه می دهد. می توانید unit test های دلخواه خود را به صورت انفرادی در view مذکور انتخاب نموده و پس از راست کلیک بر روی آن و انتخاب گزینه‌ی Run، آن ها را مجدداً اجرا نمایید.



به صورت پیش فرض، این view تمامی تست های موجود را به نمایش می گذارد. می توانید این view را طوری تنظیم نمایید که تنها تست های ناموفق (failed) را نمایش دهد.



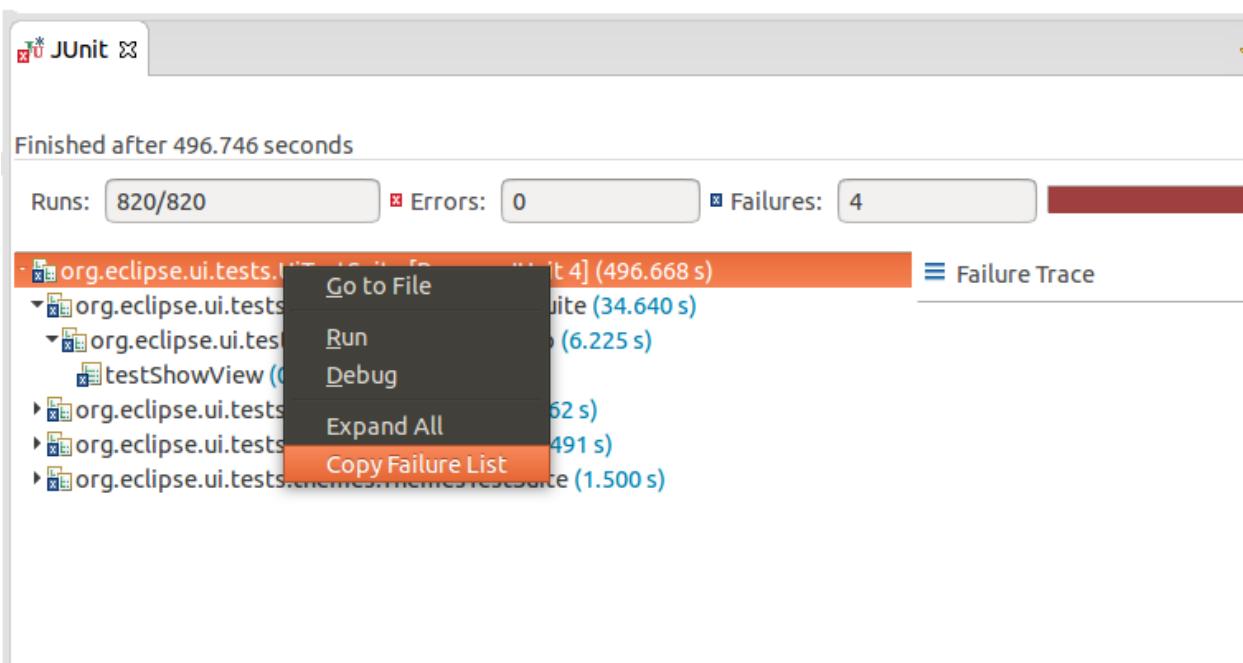
همچنین می توانید view را طوری تنظیم کنید که تنها در صورت برخورد با یک تست ناموفق فعال شود.



**توجه:** محیط برنامه نویسی Eclipse تنظیماتی مرتبط با اجرای تست ها (run configurations) است. جهت مشاهده و ویرایش آن ها می توانید به این صورت اقدام نمایید: Run > Run Configurations... menu

### 7-6-3- استخراج و بازیابی تست های ناموفق و stacktrace ها

جهت بازیابی لیست مربوط به تست ناموفق، بر روی نتیجه تست راست کلیک کرده و Copy Failure List را انتخاب نمایید. این کار سبب می شود تست های ناموفق و stack trace ها در clipboard کپی و درج شوند (stack trace) عبارت است فرایند ردیابی اجرای کدهای برنامه در بخشی از حافظه به نام .(stack).



### 7-6-4- امکان static import در فریم ورک

یک امکان است که به فیلدهایی که داخل کلاس به صورت public static تعریف شده اند، این اجازه را می دهد تا بدون ذکر اسم کلاس میزبان، بکار برد شوند.

دستورات assert فریم ورک مذکور، برای اینکه قابلیت نوشتن دستورات تست گیری (test) public static مختص و بهینه را برای توسعه دهنده فراهم کنند، معمولاً به صورت statement

تعریف می شوند. تکه کد زیر یک دستور assert را با دو حالت ممکن (با امکان static imports و یکی بدون static import) به نمایش می گذارد.

```
// without static imports you have to write the following statement
Assert.assertEquals("10 x 5 must be 50", 50, tester.multiply(10, 5));
```

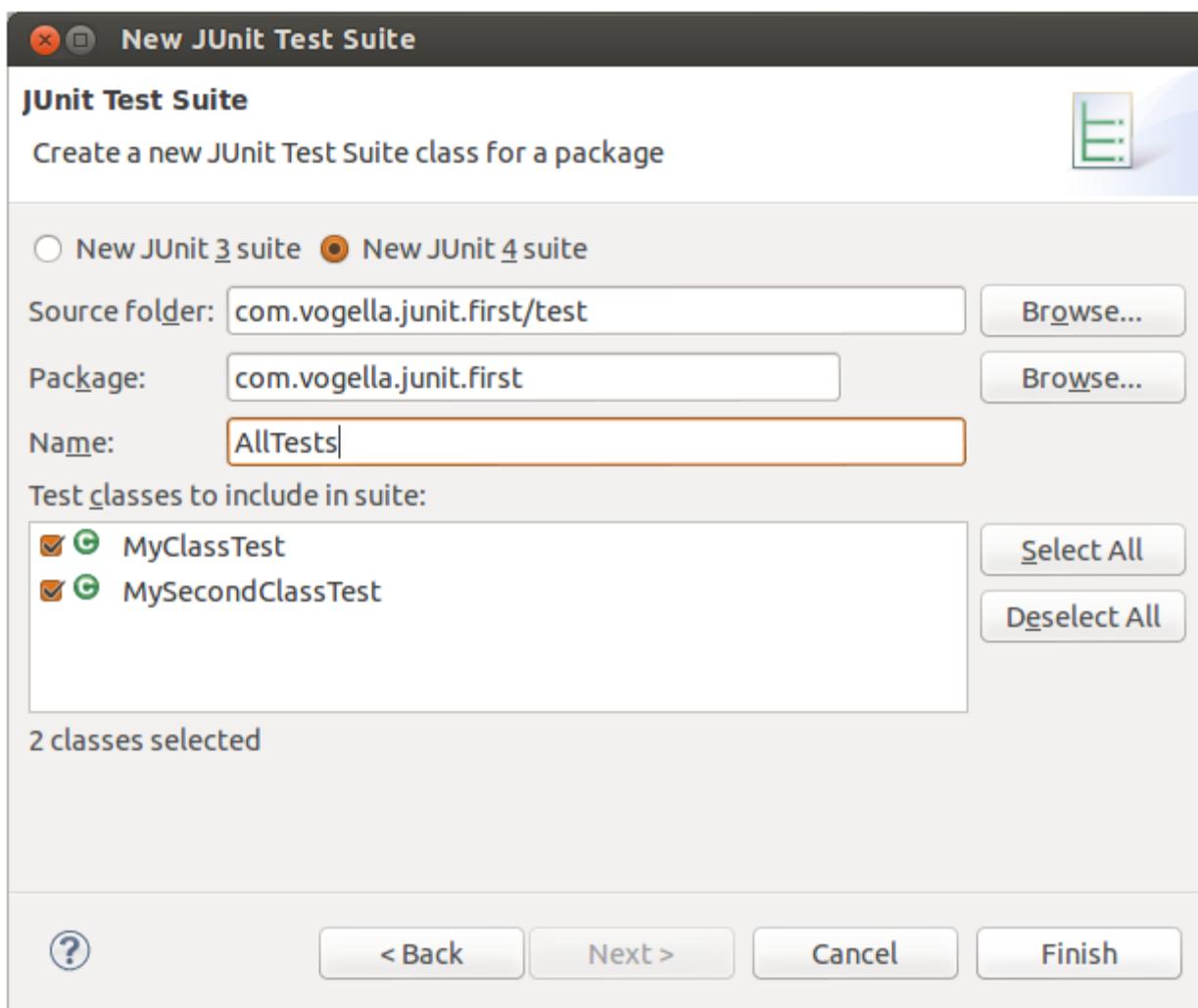
```
// alternatively define assertEquals as static import
import static org.junit.Assert.assertEquals;
```

```
// more code
```

```
// use assertEquals directly because of the static import
assertEquals("10 x 5 must be 50", 50, tester.multiply(10, 5));
```

## ۷-۶-۵-برنامه‌ی راهنمای Wizard برای ساختن مجموعه‌ی تست (test suite)

می‌توانید در محیط کاری Eclipse و با امکاناتی که در اختیار شما قرار می‌دهد، به راحتی یک test suite (مجموعه‌ی تست) ایجاد نمایید. برای نیل به این هدف، ابتدا کلاس‌های آزمایشی و تست گیری (test class) که می‌بایست در مجموعه (suite) گنجانده شود را از طریق کادر Package Explorer گزینش نموده، در این کادر بر روی آن‌ها راست کلیک نمایید و سپس مسیر رو به رو را طی کنید: .New ▶ Other... ▶ JUnit ▶ JUnit Test Suite



## آموزشگاه حکلیکر داده

### 7-6-6- تست exception و خطاهای

دارای قابلیتی محدودی است چرا که تنها قادر است یک exception (خطا) را تست کند. به منظور تست کلی exception ها، می توانید از الگوی تست گیری (test pattern) زیر استفاده نمایید.

```
try {
    mustThrowException();
    fail();
} catch (Exception e) {
    // expected
}
```

640

آدرس آموزشگاه : تهران - خیابان شریعتی - بالاتر از خیابان ملک - جنب بانک صادرات - پلاک 651 طبقه دوم - واحد 7

88146323 - 88446780 - 88146330

```
// could also check for message of exception, etc.  
}
```

## 7-6-7- تست نویسی برای افزونه ها (JUnit Plug-in Test)

توسعه دهنده می تواند با استفاده از تست های JUnit Plug-in برای افزونه های نرم افزاری خود نیز تست طراحی کند. این تست ها در بستر test runner ویژه راه اندازی شده که به یک نمونه ی مستقل VM برای اجرا در محیط Eclipse احتیاج دارد. در واقع متدهای تست گیری محصور در کلاس (test class)، داخل آن نمونه مجزا (instance) (test class) اجرا می شوند.

## 7-7- نصب JUnit

### 7-7-1- استفاده از JUnit با سیستم کامپایل Gradle

به منظور استفاده از JUnit در فایل Gradle build، لازم است کتابخانه (dependency) testCompile را به فایل build خود اضافه نمایید.

```
apply plugin: 'java'  
dependencies {  
    testCompile 'junit:junit:4.12'  
}
```

### 7-7-2- استفاده از JUnit با سیستم کامپایل Maven

به منظور استفاده از JUnit در فایل Maven build، می بایست کتابخانه (dependency) زیر را به فایل pom خود اضافه نمایید.

```
<dependency>  
    <groupId>junit</groupId>  
    <artifactId>junit</artifactId>  
    <version>4.12</version>  
</dependency>
```

### 7-7-3- استفاده از JUnit درون ساخته‌ی محیط کاری Eclipse

محیط برنامه نویسی Eclipse با یک ورژن خاص از JUnit در اختیار برنامه نویسان قرار می‌گیرد. اگر از Eclipse برای ساخت و توسعه‌ی اپلیکیشن خود بهره می‌گیرید، در آن صورت لزومی ندارد افزونه‌ی یا محتوا‌ی جدیدی از اینترنت دانلود نمایید.

### 7-7-4- دانلود کتابخانه‌ی JUnit

اگر می‌خواهید کتابخانه‌ی JUnit مورد استفاده را به صورت صریح کنترل کنید، در آن صورت کافی است فایل JUnit4.x.jar را از وب سایت زیر دانلود کنید. محتوا‌ی دانلود شده، junit-4.\*.jar را دربردارد که همان کتابخانه‌ی JUnit است. این کتابخانه را به پروژه‌ی Java خود و classpath مورد نظر اضافه نمایید.

<http://junit.org/>

### 7-8- تنظیم محیط برنامه نویسی Eclipse برای استفاده از امکان static import کتابخانه‌ی JUnits

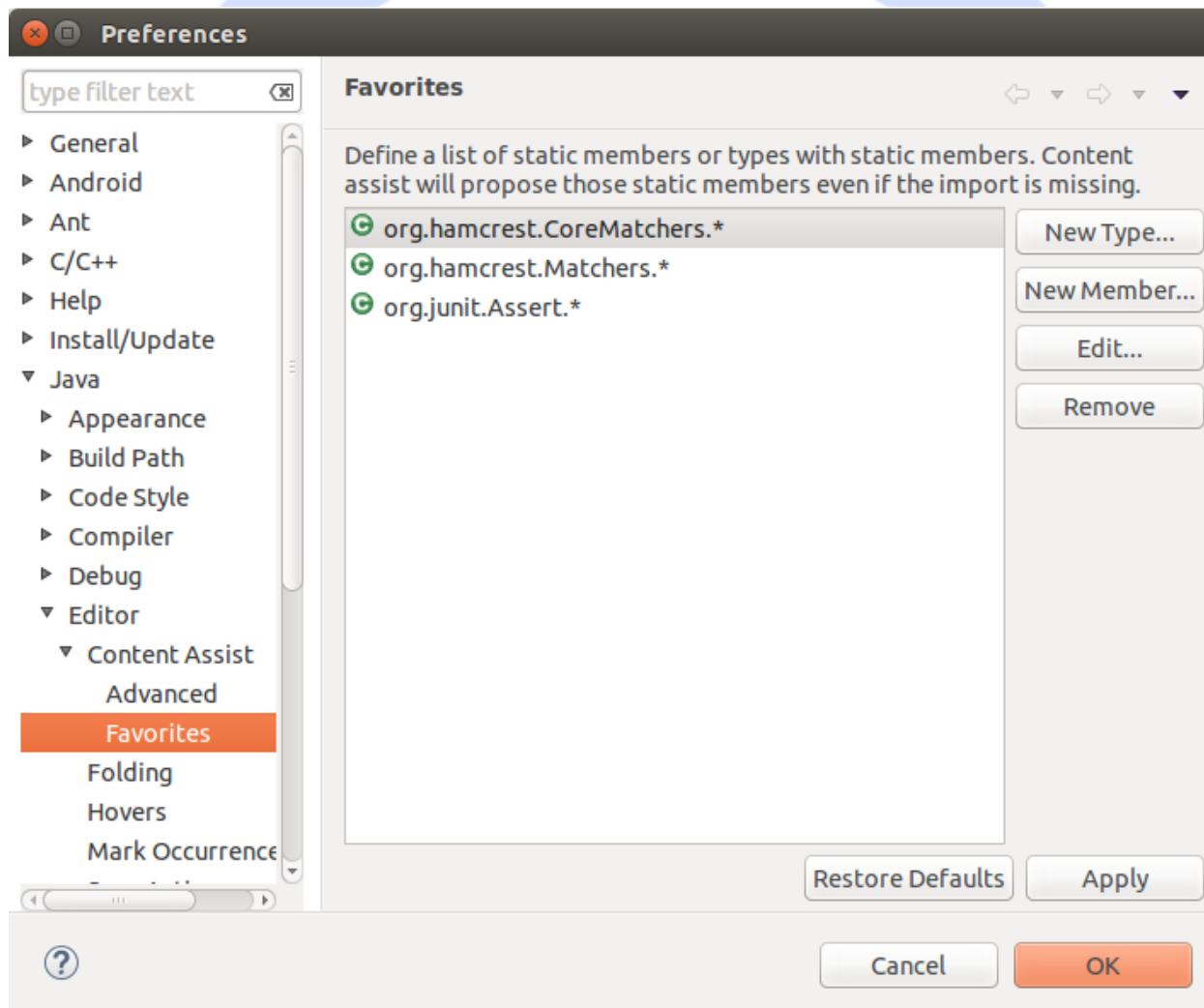
محیط کاری Eclipse به خودی خود همیشه قابلیت ایجاد دستورات static import مربوطه را ندارد. شما می‌توانید این محیط برنامه نویسی را طوری تنظیم نمایید که با استفاده از امکان static completion خود توابع معمول و پرکاربرد JUnit را وارد متن برنامه کرده و دستورات Preferences را به صورت خودکار اضافه نمایید. برای دستیابی به این هدف، ابتدا پنجره‌ی import Java را از طریق Window ▶ Preferences باز نمایید و سپس مسیر رو به رو را طی کنید:

.Editor ▶ Content Assist ▶ Favorites

با استفاده از دکمه **New Type** دستورات زیر را به آن اضافه نمایید:

- org.junit.Assert
- org.hamcrest.CoreMatchers
- org.hamcrest.Matchers

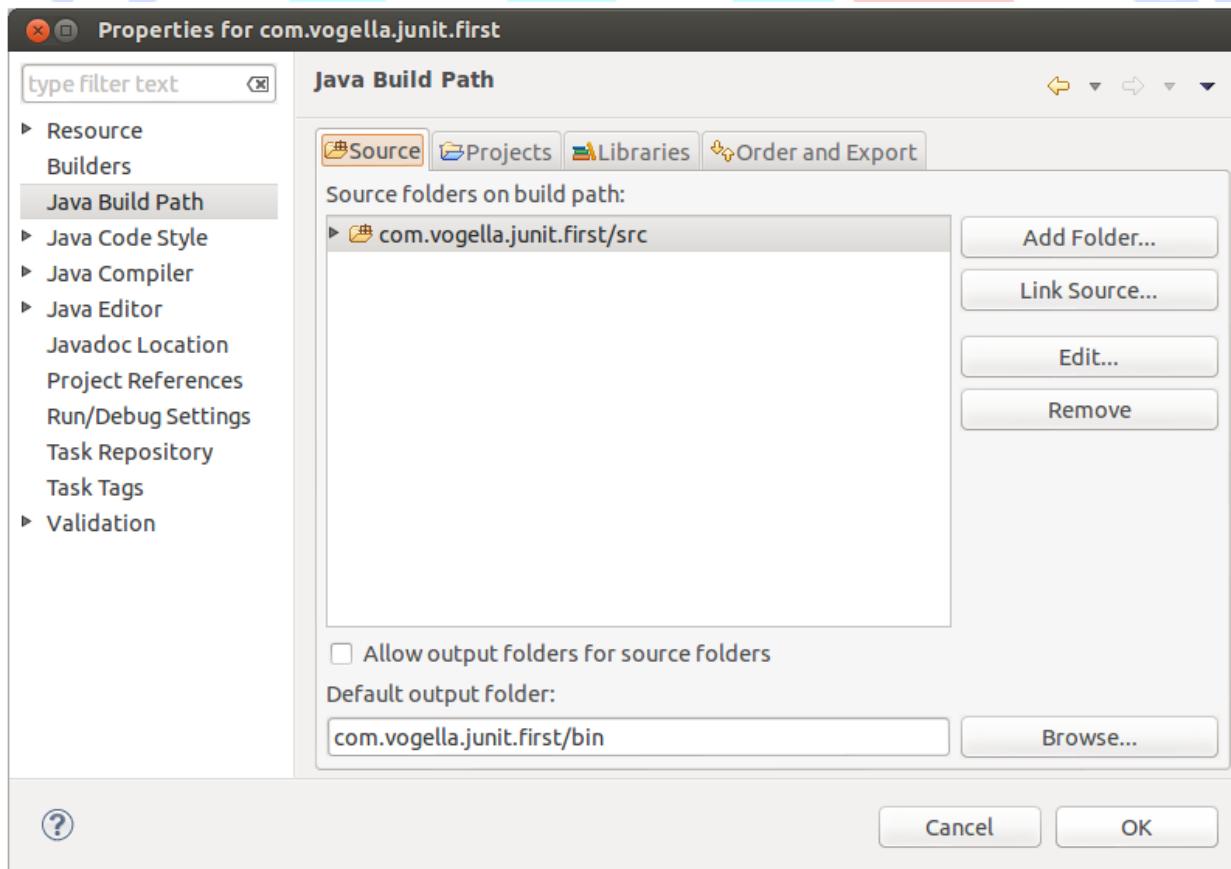
با این کار متدهای assertEquals، assertFalse و assertTrue را مستقیما در Content Assists قابل دسترسی و آماده استفاده می نمایید.



اکنون شما می توانید با استفاده از Content Assists (که با فشردن کلیدهای Ctrl+Space) طور همزمان به سرعت در اختیار شما قرار می گیرد) به راحتی متاد و import را اضافه نمایید.

## تمرین: استفاده از JUnit آماده سازی پروژه

یک پروژه می باشد به نام com.vogella.junit.first ایجاد نمایید. سپس یک پوشه می باشد به نام test در پروژه مذکور ایجاد کنید. برای این منظور بر روی پروژه مورد نظر راست کلیک کرده، Source Properties را انتخاب نمایید و این مسیر را طی کنید: Java ▶ Build Path. حال بر روی تب کلیک نموده و آن را انتخاب کنید.



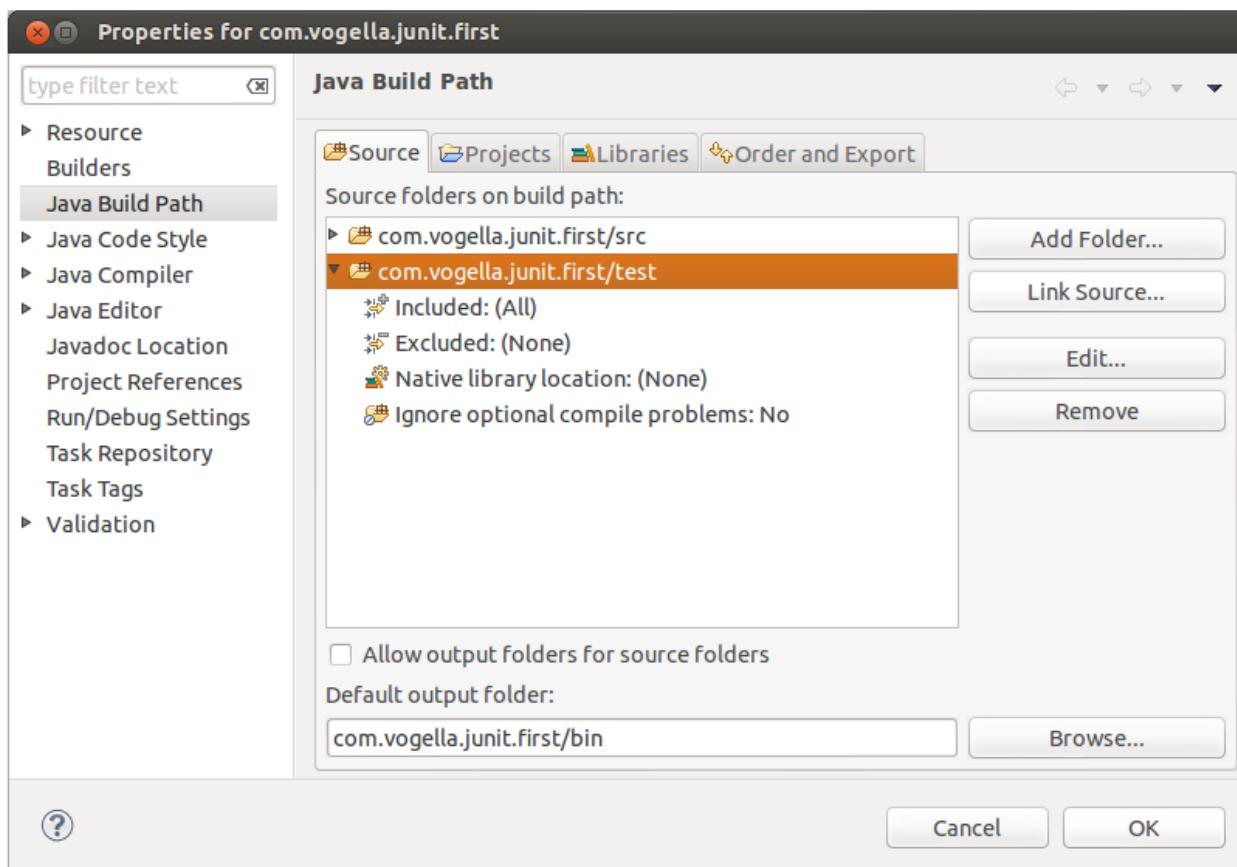
دکمه **ى** Add Folder را کلیک کنید. پس از آن، دکمه **ى** Create New Folder را فشار دهید. اسم پوشه **را** test انتخاب نمایید.



645

آدرس آموزشگاه : تهران - خیابان شریعتی - بالاتر از خیابان ملک - جنب بانک صادرات - پلاک 651 طبقه دوم - واحد 7

88146323 - 88446780 - 88146330



**توجه:** در صورت تمایل می توانید با راست کلیک بر روی پروژه و انتخاب **New > Source Folder** یک پوشه ای اصلی جدید نیز اضافه نمایید.

## 7-8-1 ساخت یک کلاس Java

در پوشه ای `src`, پکیج `com.vogella.junit.first` را ایجاد کرده و سپس کلاس زیر را به آن اضافه نمایید.

```
package com.vogella.junit.first;
public class MyClass {
    public int multiply(int x, int y) {
        // the following is just an example
        if (x > 999) {
            throw new IllegalArgumentException("X should be less than 1000");
        }
        return x / y;
    }
}
```

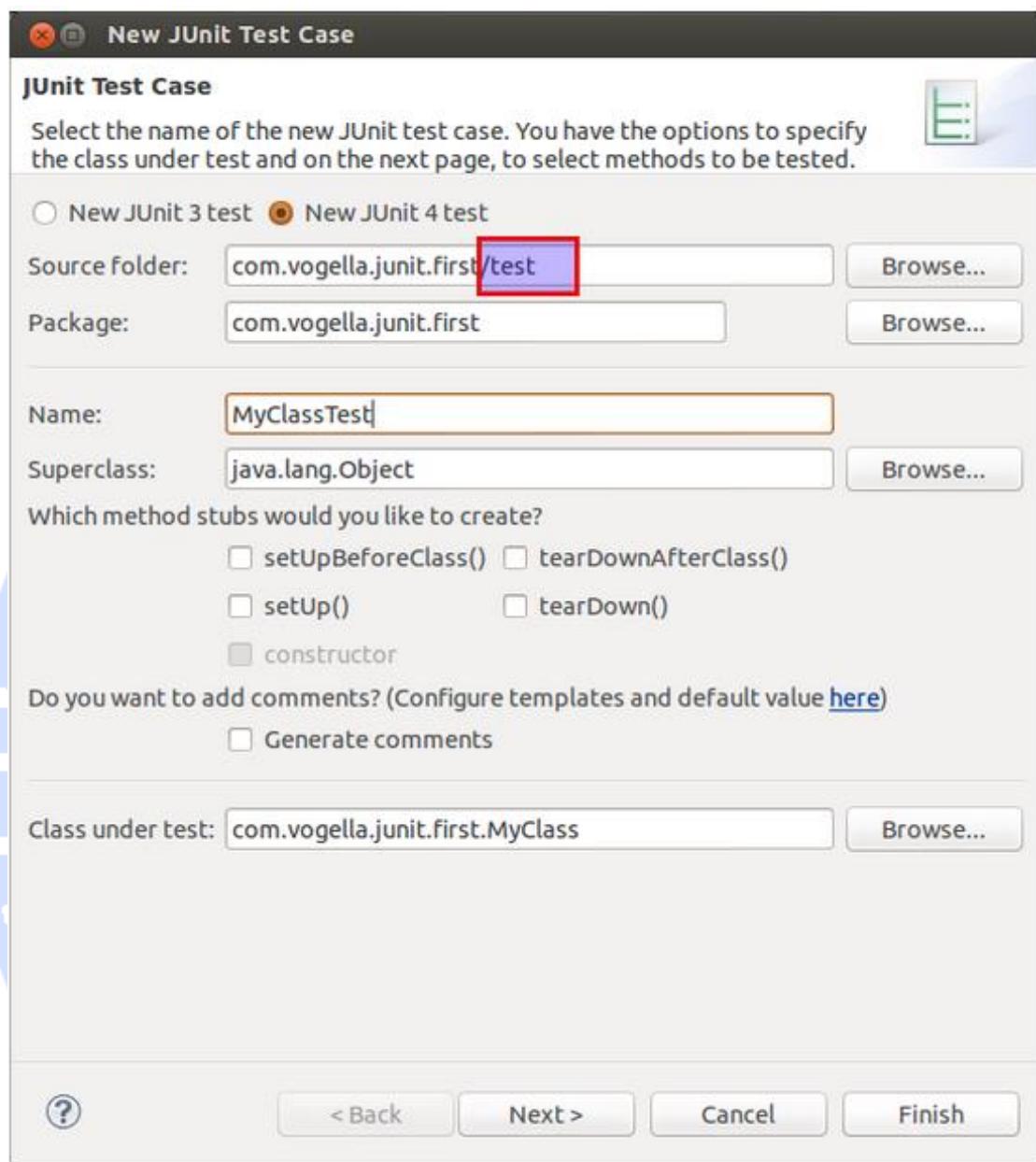
```
{
}
```

## 7-8-2-ساخت و طراحی یک تست JUnit

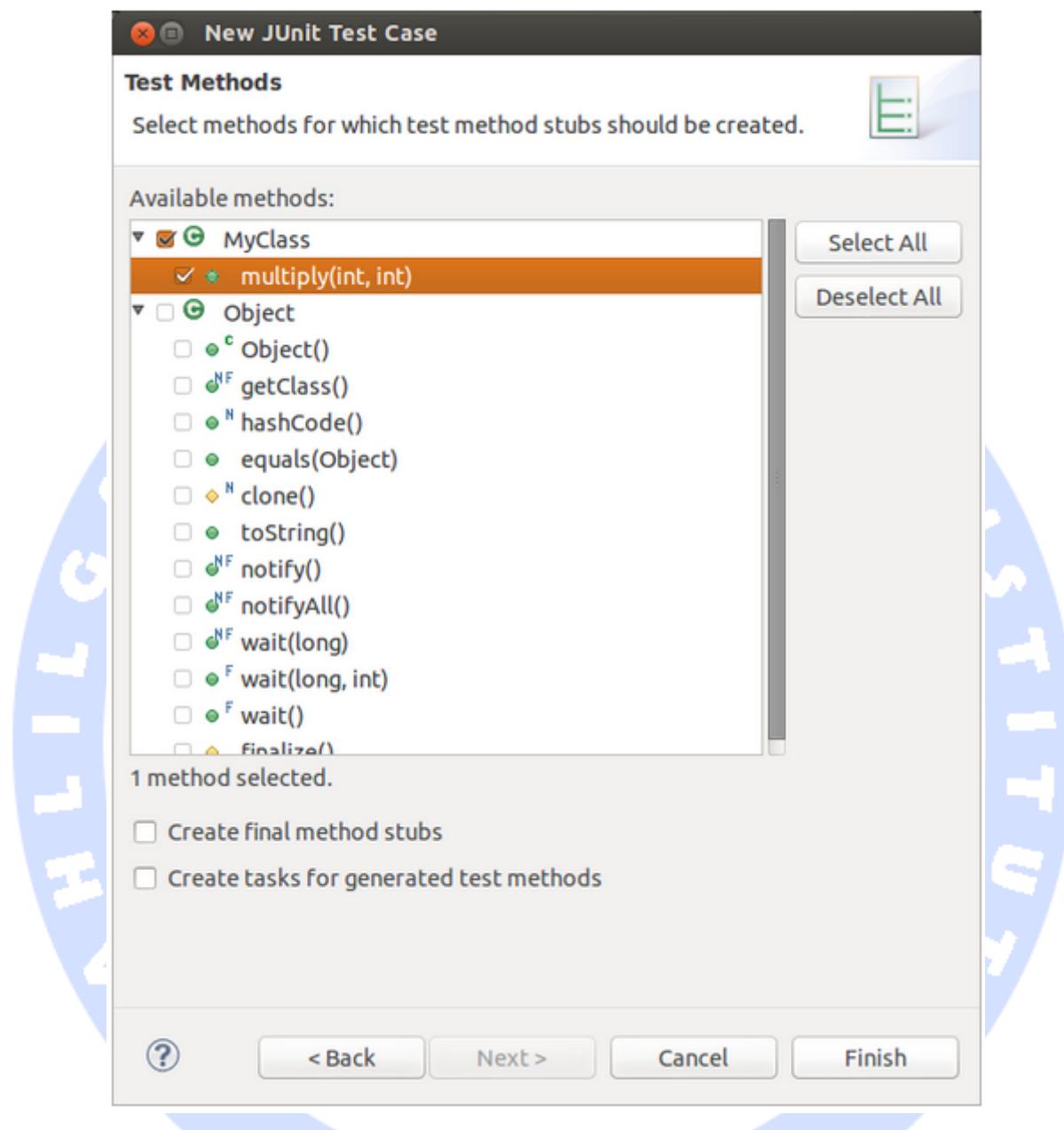
بر روی کلاس جدید، داخل کادر Package Explorer، راست کلیک نموده و سپس این مسیر را طی نمایید: **New ▶ JUnit Test Case**

در برنامه ی راهنمای نصب (ویزارد) که در زیر مشاهده می کنید، حتما دکمه ی رادیویی **New JUnit 4 test** را فعال کرده و سپس پوشه ی **source test** تنظیم نمایید تا کلاس تست در این پوشه ایجاد شود.

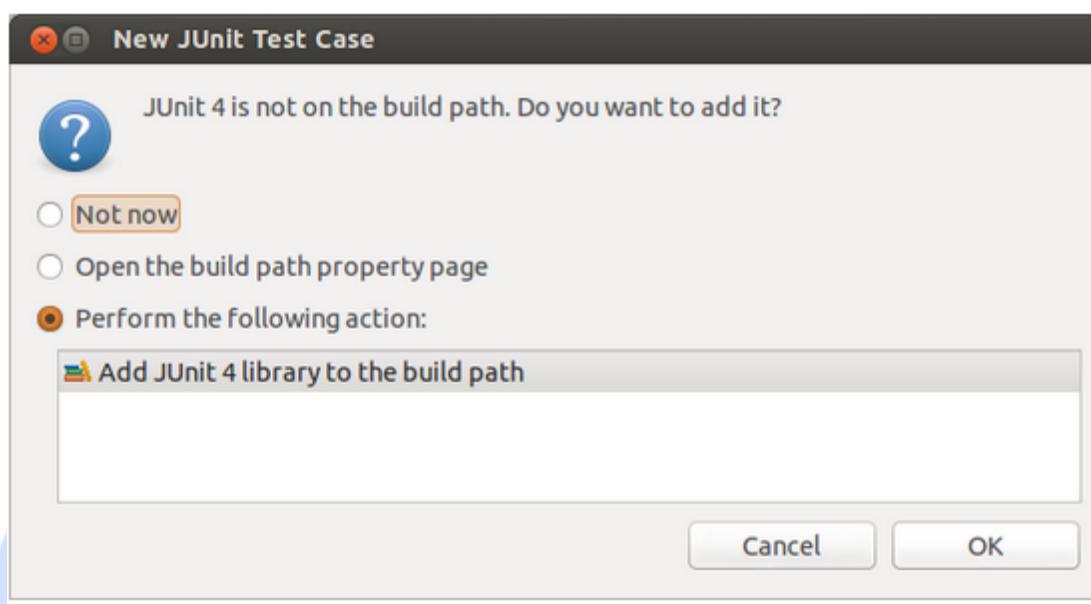




بر روی دکمه **Next** کلیک کرده و متدهایی که می خواهید تست شوند را انتخاب نمایید.



اگر کتابخانه‌ی JUnit در classpath پروژه‌ی جاری قید نشده بود، در آن صورت محیط کاری از شما می‌خواهد که حتماً آن را اضافه نمایید. با این کار شما در واقع JUnit و امکانات آن را به پروژه‌ی خود اضافه می‌نمایید.

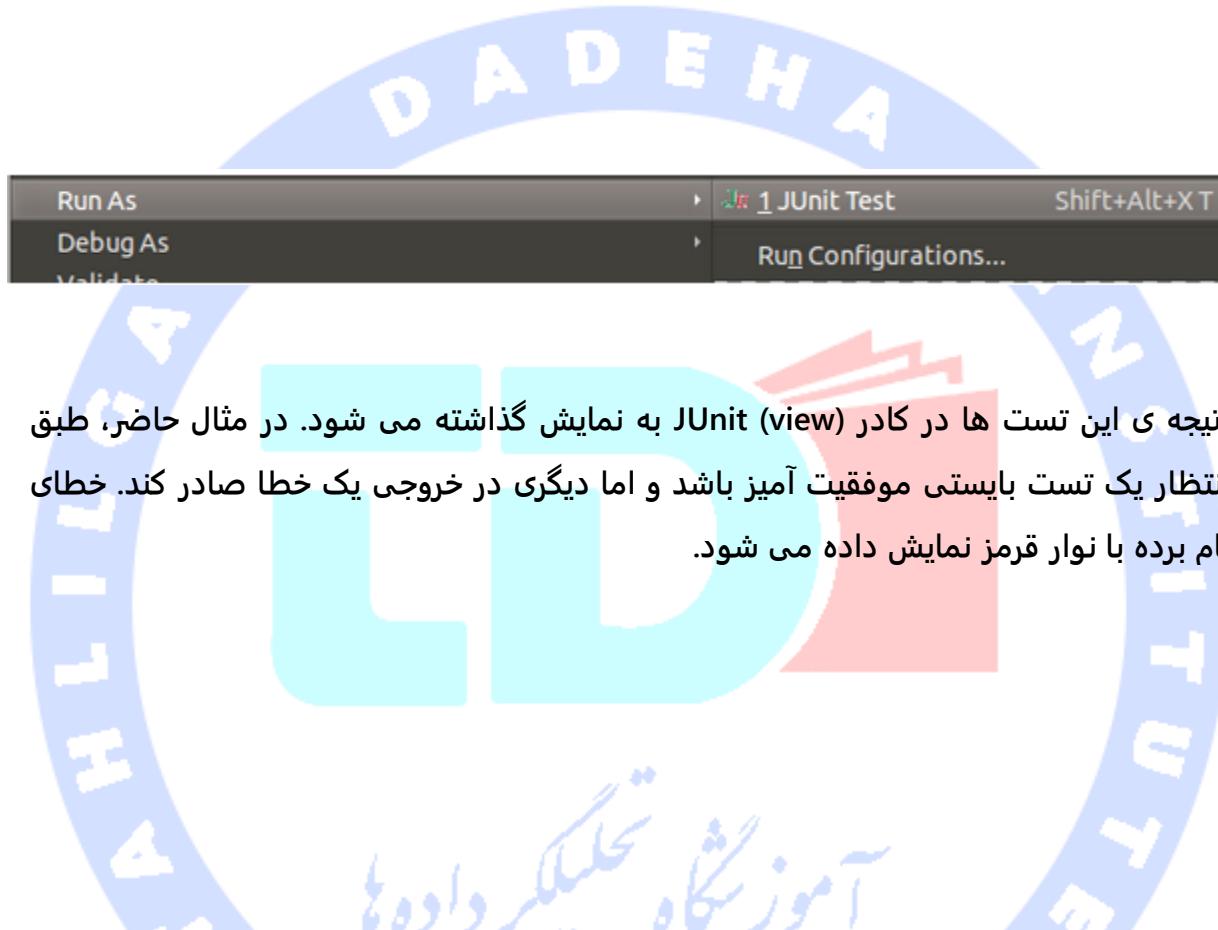


حال تست مورد نظر را با کد و محتوا زیر ایجاد نمایید.

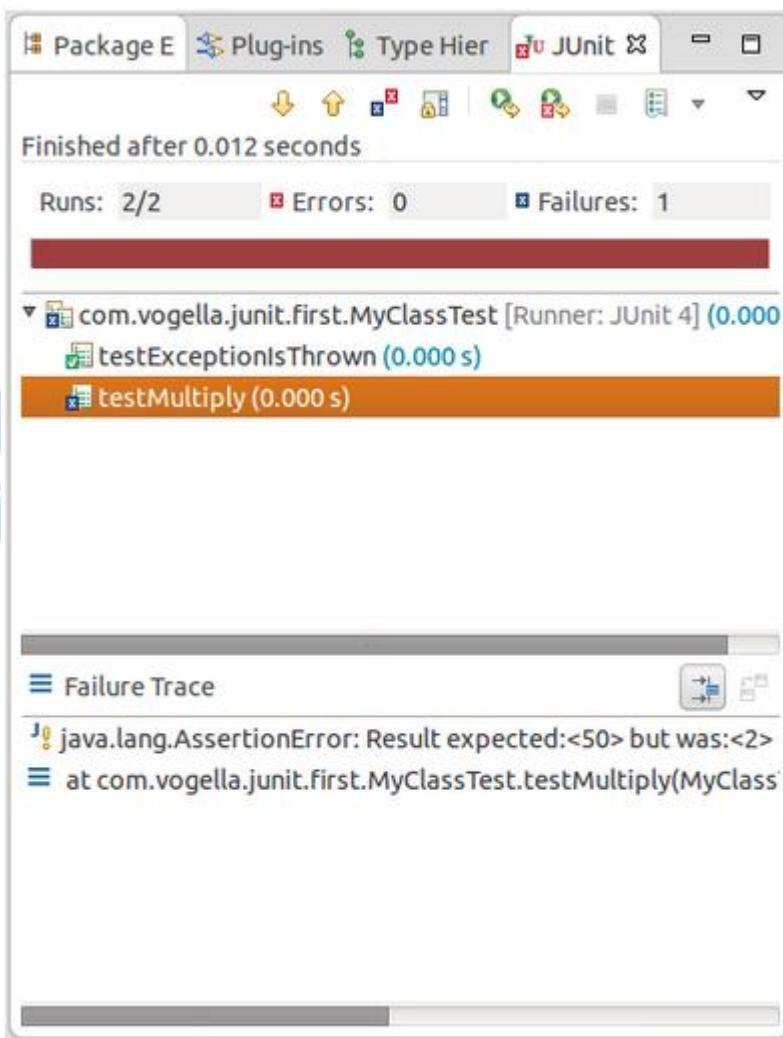
```
package com.vogella.junit.first;
import static org.junit.Assert.assertEquals;
import org.junit.AfterClass;
import org.junit.BeforeClass;
import org.junit.Test;
public class MyClassTest {
    @Test(expected = IllegalArgumentException.class)
    public void testExceptionThrown() {
        MyClass tester = new MyClass();
        tester.multiply(1000, 5);
    }
    @Test
    public void testMultiply() {
        MyClass tester = new MyClass();
        assertEquals("10 x 5 must be 50", 50, tester.multiply(10, 5));
    }
}
```

## 7-8-2- اجرای تست در محیط برنامه نویسی Eclipse

بر روی کلاس تست گیری (test class) جدید راست کلیک کرده و سپس گزینه **JUnit** را انتخاب نمایید.



نتیجه‌ی این تست‌ها در کادر **JUnit** (view) به نمایش گذاشته می‌شود. در مثال حاضر، طبق انتظار یک تست باستثنی موفقیت آمیز باشد و اما دیگری در خروجی یک خطأ صادر کند. خطای نام بردۀ با نوار قرمز نمایش داده می‌شود.



تست حاضر به این خاطر ناموفق است که کلاس (ماشین حساب) در انجام عملیات مورد انتظار (ضرب) با مشکل مواجه می شود. در واقع به جای انجام عملیات ضرب، دو عدد را بر هم تقسیم می کند. حال اشکال فنی برنامه را برطرف نموده و بار دیگر تست را اجرا کنید تا نوار سبز در خروجی تست به نمایش گذاشته شود.

## 7- تنظیمات و امکانات پیشرفته‌ی JUnit

## 7-9-1- اجرای تست های دارای پارامتر مشخص (Parameterized test)

JUnit این امکان را برای شما فراهم می آورد تا پارامترهای دلخواه را در کلاس های تست گیری مورد استفاده قرار دهید. کلاس مورد نظر می تواند شامل یک متده است گیری بوده و این متده هر بار با پارامترهای ارائه شده ی مختلف اجرا شود.

بایستی کلاسی که قرار است پارامترهای مختلف به آن ارسال شوند را با دستور `@RunWith(Parameterized.class)`

چنین کلاسی (کلاس تست گیری) باید یک متده static در خود داشته که با دستور `@Parameters` نشانه گذاری شده باشد. این متده یک مجموعه آرایه (array collection) تولید کرده و در خروجی `test` بازمی گرداند. تک تک آیتم ها در این آرایه به عنوان پارامتر ورودی به متده است گیری (method) حاضر در این کلاس ارسال می شوند.

می توانید فیلد های public را با دستور `@Parameter` نشانه گذاری نمایید و بدین سیله مقادیر تستی (مقادیر یا پارامترهای آزمایشی) را داخل تست تزریق نمایید.

کد زیر نمونه ای از یک تست که پارامترهایی از بیرون به آن تزریق می شوند را به نمایش می گذارد. تست حاضر متده multiply() از کلاس MyClass را که یک کلاس دورنی و nested داخل کلاس میزبان می باشد را با ارسال پارامتر به آن، به منظور تست، آزمایش می نماید.

```
package testing;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.junit.runners.Parameterized;
import org.junit.runners.Parameterized.Parameters;
import java.util.Arrays;
import java.util.Collection;
import static org.junit.Assert.assertEquals;
import static org.junit.runners.Parameterized.*;
@RunWith(Parameterized.class)
public class ParameterizedTestFields {
    // fields used together with @Parameter must be public
    @Parameter
    public int m1;
    @Parameter (value = 1)
```

```

public int m2;
// creates the test data
@Parameters
public static Collection<Object[]> data() {
    Object[][] data = new Object[][] { { 1, 2 }, { 5, 3 }, { 121, 4 } };
    return Arrays.asList(data);
}
@Test
public void testMultiplyException() {
    MyClass tester = new MyClass();
    assertEquals("Result", m1 * m2, tester.multiply(m1, m2));
}
// class to be tested
class MyClass {
    public int multiply(int i, int j) {
        return i * j;
    }
}
}

```

یا می توانید به جای استفاده از دستور `@Parameter`، تابع سازنده (constructor) را بکار ببرید. در این تابع مقادیر مورد استفاده در هر تست نگهداری می شود. لازم به ذکر است که تعداد المان های موجود در آرایه که متده مورد نظر (متده که با دستور فوق نشانه گذاری شده باشد) ارائه می دهد، بایستی با تعداد پارامترهای حاضر در تابع سازنده ی کلاس برابر باشد. در واقع به ازای هر پارامتر یک کلاس مستقل ایجاد شده و سپس مقادیر آزمایشی به واسطه ی تابع سازنده می شوند.

به کلاس مورد نظر فرستاده (constructor)

```

package de.vogella.junit.first;
import static org.junit.Assert.assertEquals;
import java.util.Arrays;
import java.util.Collection;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.junit.runners.Parameterized;
import org.junit.runners.Parameterized.Parameters;
@.RunWith(Parameterized.class)
public class ParameterizedTestUsingConstructor {
    private int m1;
    private int m2;
    public ParameterizedTestUsingConstructor(int p1, int p2) {
        m1 = p1;
        m2 = p2;
    }
    // creates the test data
    @Parameters
    public static Collection<Object[]> data() {
        Object[][] data = new Object[][] { { 1, 2 }, { 5, 3 }, { 121, 4 } };
        return Arrays.asList(data);
    }
}

```

```

        return Arrays.asList(data);
    }
    @Test
    public void testMultiplyException() {
        MyClass tester = new MyClass();
        assertEquals("Result", m1 * m2, tester.multiply(m1, m2));
    }
    // class to be tested
    class MyClass {
        public int multiply(int i, int j) {
            return i *j;
        }
    }
}

```

حال اگر این کلاس تست گیری (test class) را اجرا کنید، متدها در آن هر بار با پارامتر تعریف شده به صورت جداگانه اجرا می شود. به عنوان مثال، در نمونه‌ی فوق متدهای تست گیری مورد نظر سه بار اجرا می شود.

با مراجعه به آدرس اینترنتی <https://github.com/Pragmatists/JUnitParams> می توانید با روشی که نوشتند آن به مراتب آسان و معطف‌تر هست، آشنا شوید.

## 7-9-2- دستور (@Rule annotation)

Rule های کتابخانه‌ی JUnit به شما این امکان را می دهند تا رفتار و قابلیت‌های مورد نظر را به تک تک تست‌های داخل کلاس تست گیری (test class) اختصاص داده و اعمال کنید. می توانید فیلد هایی که از جنس TestRule هستند را با `@Rule` نشانه گذاری نمایید. همچنین می توانید آبجکت هایی ایجاد کنید که قابلیت تنظیم و استفاده از آن‌ها در متدهای تست گیری وجود داشته باشد. با این کار به میزان انعطاف پذیری تست شما افزوده می شود. به طور مثال، می توانید مشخص کنید که به هنگام اجرای کد آزمایشی، انتظار دارید که کدام پیغام خطا (exception message) نمایش داده شود.

```

package de.vogella.junit.first;
import org.junit.Rule;
import org.junit.Test;
import org.junit.rules.ExpectedException;
public class RuleExceptionTesterExample {
    @Rule
    public ExpectedException exception = ExpectedException.none();
}

```

```

    @Test
    public void throwsIllegalArgumentExceptionIfIconIsNull() {
        exception.expect(IllegalArgumentException.class);
        exception.expectMessage("Negative value not allowed");
        ClassToBeTested t = new ClassToBeTested();
        t.methodToBeTest(-1);
    }
}

```

JUnit از قبل تعداد زیادی از rule ها را پیاده سازی نموده و آن ها را به صورت آماده در اختیار توسعه دهنده قرار می دهند. به طور مثال، کلاس TemporaryFolder این امکان را به شما می دهد تا فایل ها و پوشه هایی تنظیم کنید که خود به خود پس از هر بار اجرای تست حذف می شوند. نمونه‌ی زیر استفاده از پیاده سازی TemporaryFolder را به نمایش می‌گذارد.

```

package de.vogella.junit.first;
import static org.junit.Assert.assertTrue;
import java.io.File;
import java.io.IOException;
import org.junit.Rule;
import org.junit.Test;
import org.junit.rules.TemporaryFolder;
public class RuleTester {
    @Rule
    public TemporaryFolder folder = new TemporaryFolder();
    @Test
    public void testUsingTempFolder() throws IOException {
        File createdFolder = folder.newFolder("newfolder");
        File createdFile = folder.newFile("myfilefile.txt");
        assertTrue(createdFile.exists());
    }
}

```

## آموزشگاه تحلیلکردن و آنلاین

### 7-9-3- طراحی و تنظیم rule های اختصاصی برای JUnit

جهت نوشتن rule های اختصاصی و دلخواه خود، شما می بایست اینترفیس TestRule را پیاده سازی نمایید. این اینترفیس، متد apply(Statement, Description) را در اختیار توسعه دهنده قرار می دهد که خروجی آن نمونه‌ی ای از Statement می باشد. پارامتر Statement نشانگر تست های موجود در بستر JUnit runtime () است و Statement#evaluate() نیز تابعی است که این

تست ها را به هنگام runtime ، اجرا می کند. پارامتر Description، توصیف گر تست های فردی بوده و به شما اجازه می دهد تا از طریق reflection اطلاعات مرتبط با تست ها را بخوانید.

نمونه ی زیر یک مثال ساده از افزودن دستور log به اپلیکیشن اندرویدی، قبل و بعد از اجرای تست می باشد.

```
 package testing.android.vogella.com.AsyncTask;
 import android.util.Log;
 import org.junit.rules.TestRule;
 import org.junit.runner.Description;
 import org.junit.runners.model.Statement;
 public class MyCustomRule implements TestRule {
     private Statement base;
     private Description description;
     @Override
     public Statement apply(Statement base, Description description) {
         this.base = base;
         this.description = description;
         return new MyStatement(base);
     }
     public class MyStatement extends Statement {
         private final Statement base;
         public MyStatement(Statement base) {
             this.base = base;
         }
         @Override
         public void evaluate() throws Throwable {
             System.out.println("MyCustomRule " + description.getMethodName());
             Log.w("MyCustomRule",description.getMethodName() + " Started");
             try {
                 base.evaluate();
             } finally {
                 Log.w("MyCustomRule",description.getMethodName() + " Finished");
             }
         }
     }
 }
```

به منظور استفاده از این rule، کافی است یک فیلد که با @Rule نشانه گذاری شده باشد به کلاس تست گیری (test class) خود اضافه نمایید.

```
 @Rule
 public MyCustomRule myRule = new MyCustomRule();
```

## Category-7-9-4 ها و دسته بندی تست ها

می توان تست ها را دسته بندی نموده و یا آن ها را بر اساس اسم annotation ها، داخل دسته ای مورد نظر گنجاند یا از دسته ای حذف کرد. مثال زیر بر اساس نکات و رهنمودهای منتشر شده در ویرایش JUnit 4.8 می باشد.

```
public interface FastTests { /* category marker */ }
public interface SlowTests { /* category marker */ }
public class A {
    @Test
    public void a() {
        fail();
    }
    @Category(SlowTests.class)
    @Test
    public void b() {
    }
}
@Category({ SlowTests.class, FastTests.class })
public class B {
    @Test
    public void c() {
    }
}
@RunWith(Categories.class)
@IncludeCategory(SlowTests.class)
@SuiteClasses({ A.class, B.class })
// Note that Categories is a kind of Suite
public class SlowTestSuite {
    // Will run A.b and B.c, but not A.a
}
@RunWith(Categories.class)
@IncludeCategory(SlowTests.class)
@ExcludeCategory(FastTests.class)
@SuiteClasses({ A.class, B.class })
// Note that Categories is a kind of Suite
public class SlowTestSuite {
    // Will run A.b, but not A.a or B.c
}
```

آموزشگاه تحلیلکر داده

## 7-10- ایجاد آبجکت های ساختگی یا شبیه سازی رفتار آبجکت/Mocking

برای اجرای تست بر روی بخش های مختلف پروژه‌ی نرم افزاری که همان unit testing نیز خوانده می‌شود، گاه لازم است از object mocking استفاده نمایید. در این سناریو آبجکت واقعی با یک آبجکت ساختگی که رفتار و عملکرد آبجکت مورد نظر را شبیه سازی می‌کند، در طول اجرای تست جایگزین می‌شود.

فریم ورک های متعددی برای mocking (جهت تست نرم افزار) وجود دارد. به منظور کسب اطلاعات بیشتر در مورد فریم ورک های شبیه سازی و تست نرم افزار می‌توانید به آموزش Mockito تحت آدرس <http://www.vogella.com/tutorials/Mockito/article.html> یا EasyMock تحت آدرس <http://www.vogella.com/tutorials/EasyMock/article.html> مراجعه فرمایید.



مبحث حاضر به شما آموزش می‌دهد چگونه می‌توانید برای اپلیکیشن های اندرویدی خود instrumentation طراحی نموده و از عملکرد صحیح بخش های مختلف پروژه خود اطمینان حاصل نمایید. سپس برای شما شرح می‌دهد چگونه این تست ها را در محیط کاری Android Studio و با استفاده از سیستم کامپایل Gradle اجرا نمایید.

برای یادگیری این بخش لازم است با برنامه نویسی اندروید آشنایی کافی داشته باشد.

## 10-7-مقدمه ای بر تست اپلیکیشن های اندرویدی

### 10-7-1-تست پروژه های اندرویدی

اپلیکیشن های اندرویدی بر روی دستگاه هایی اجرا می شوند که دارای حافظه و باتری محدودی بوده و پردازندۀ ضعیفی دارند. رفتار اپلیکیشن همچنین بر عوامل خارجی همچون اتصال به اینترنت، استفاده ای کلی از سیستم وغیره ... بستگی دارد. با توجه به آنچه گفته شد، اشکال زدایی، تست و بهینه سازی اپلیکیشن های اندرویدی بسیار مهم می باشد. انتخاب بخش هایی که باید تست شوند نیز حائز اهمیت بوده، به شما کمک می کند تا اپلیکیشن اندرویدی خود را بهبود ببخشید و در آینده عملیات تعمیر و نگهداشت آن را آسان می سازد.

از آنجایی که امکان تست اپلیکیشن های اندرویدی بر روی تمامی دستگاه ها (با تنظیمات و سخت افزار مختلف) وجود ندارد، توصیه می شود تست های نرم افزاری خود را بر روی دستگاه هایی با پیکربندی و سخت افزار معمول اجرا نمایید. لازم است اپلیکیشن خود را حداقل یکبار بر روی دستگاهی با پایین ترین قدرت سخت افزاری تست نمایید. هرچند بد نیست که اپلیکیشن مورد نظر را بر روی دستگاه هایی که بالاترین قدرت سخت افزاری را دارند نیز تست نمایید. برای مثال اپلیکیشن خود را بر دستگاه هایی که چگالی پیکسلی و وضوح تصویری بالایی دارند، تست کرده و مطمئن شوید که برنامه ی تحت موبایل شما با این پیکربندی نیز کاملا سازگار است.

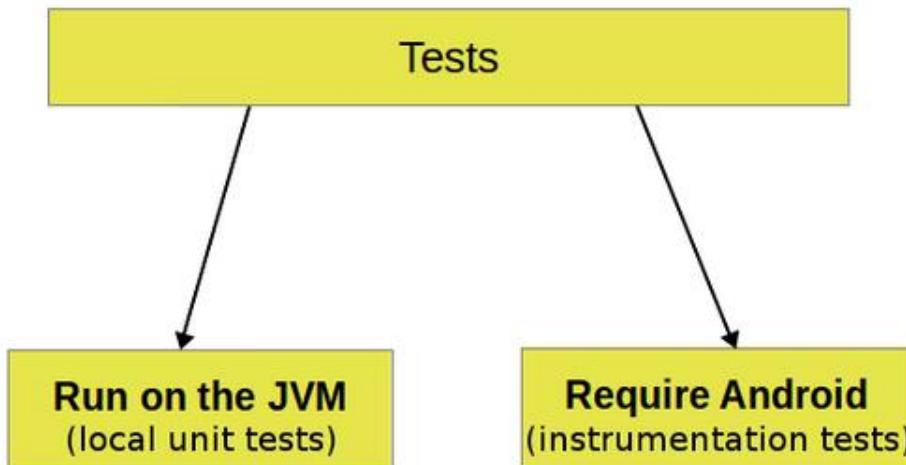
Unit testing اپلیکیشن های اندرویدی می تواند به گروه های زیر تقسیم شود:

- Local unit test (تست در بستر دستگاه مجازی جاوا) - تست هایی که می توان آن ها را در بستر JVM اجرا کرد در اصطلاح تست های local خوانده می شوند. تا حد امکان بهتر است از تست های local برای تست نرم افزار خود بهره بگیرید. تست های local به این علت که بر روی JVM اجرا می شوند، از لحاظ زمانی به خصوص نسبت به تست

هایی که بر روی دستگاه واقعی اندروید اجرا می شوند، سریع تر به انجام می رسد.

- تست بر روی دستگاه واقعی اندروید (Instrumented unit test) – تست هایی که بر روی API یک سیستم واقعی اندروید انجام می شود. اگر می خواهید کدی را امتحان کنید که به و توابع کتابخانه ای اندروید احتیاج دارد، در آن صورت لازم است این تست ها را بر روی دستگاه اندرویدی اجرا کنید. متاسفانه این امر سبب می شود زمان اجرای تست طولانی تر شود.

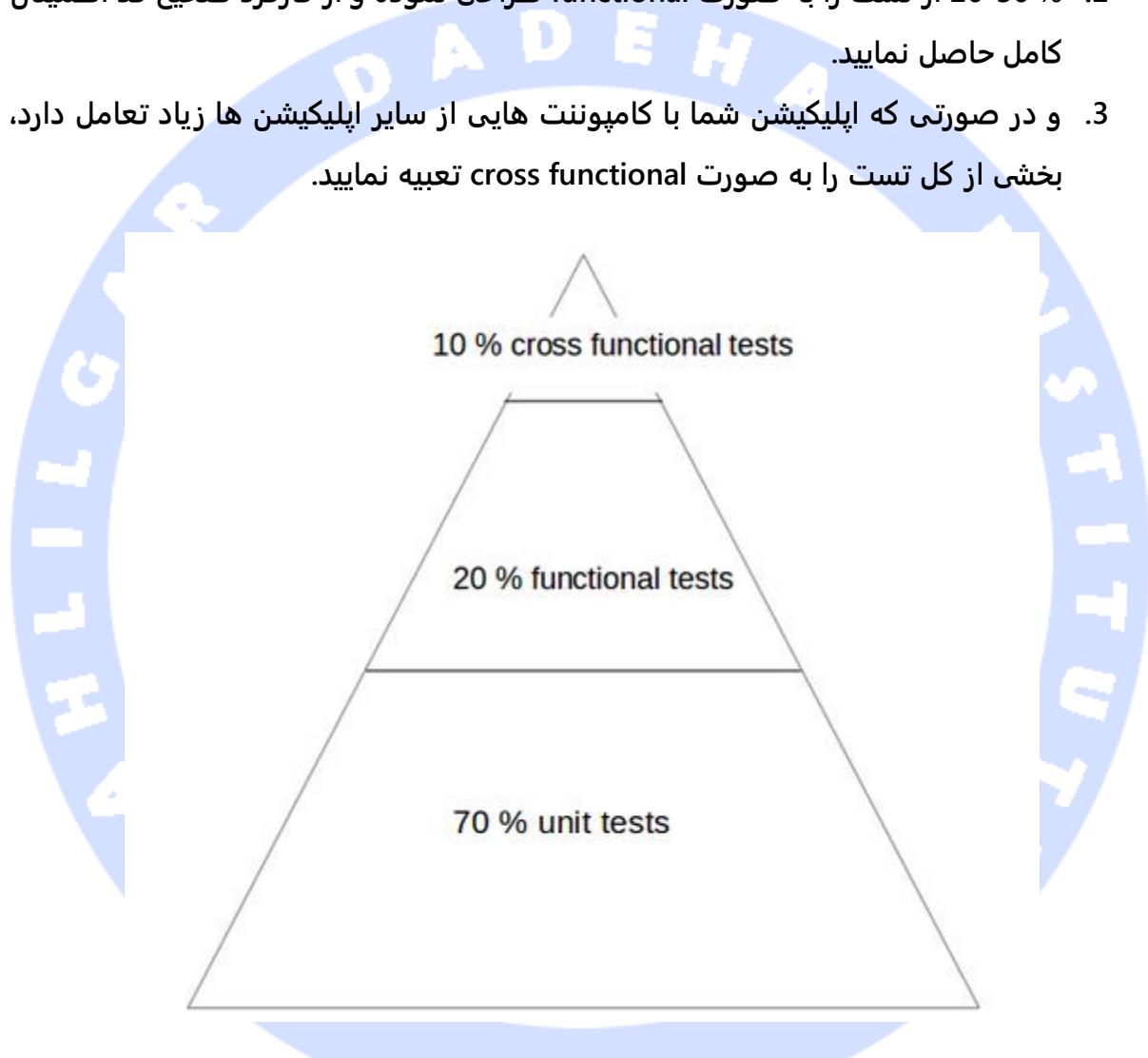
## Categories of Android tests



**7-10-7-بخش هایی که در تست نرم افزاری، تمرکز بر روی آن قرار می گیرد**  
توصیه می شود در تست نرم افزار تمرکز خود را بیشتر بر روی آزمایش و کسب اطمینان از عملکرد صحیح منطق (logic) اپلیکیشن قرار دهید.

توصیه می شود تست اپلیکیشن خود را بر اساس قاعده‌ی زیر طراحی نمایید:

1. 70-80 % از کل تست را به unit test اختصاص دهید تا بدین وسیله stability و ثبات code base تضمین شود.
2. 20-30 % از تست را به صورت functional طراحی نموده و از کارکرد صحیح کد اطمینان کامل حاصل نمایید.
3. و در صورتی که اپلیکیشن شما با کامپوننت هایی از سایر اپلیکیشن ها زیاد تعامل دارد، بخشی از کل تست را به صورت cross functional تعییه نمایید.



در تست اپلیکیشن باید به این امر نیز توجه داشته باشید که آیا تنها خود اپلیکیشن مورد آزمایش قرار می گیرد یا رابطه‌ی (integration) اپلیکیشن با اپلیکیشن های دیگر نیز مورد بررسی قرار گرفته و آزمایش می شود. در صورتی که تنها خود اپلیکیشن مورد آزمایش است، می توانید

از فریم ورک های اجرای تست استفاده نمایید که تنها به دانش جزئی از اپلیکیشن ها برای مثال view ID ها احتیاج است.

### 7-10-3-پیش شرط های تست گیری (testing preconditions)

بد نیست در تست اپلیکیشن های اندرویدی خود متذ به نام () testPreconditions داشته باشید که پیش شرط تمامی تست ها را بررسی می کند. چنانچه این متذ در انجام وظیفه ی خود موفق نباشد، بلافاصله پی می برید که فرض یا پیش شرط های دیگر تست ها نیز نقض شده است.

### 7-10-4-ATSL و تست اپلیکیشن با ابزار JUnit

Android Testing Support Library (ATSL) به شما اجازه می دهد تا اپلیکیشن خود را با یک سازگار با 4 JUnit مورد آزمایش قرار دهید. می توانید unit test runner های اپلیکیشن خود را بر روی JVM و به صورت محلی اجرا نمایید یا آن را در بستر دستگاه واقعی اندروید (Android runtime) راه اندازی کنید. علاوه بر آن، Google یک فریم ورک اجرای تست بر روی UI به نام Espresso تعییه نموده که به شما امکان توسعه ی تست های رابط کاربری را می دهد.

برای فراهم آوردن امکان تست نرم افزار در محیط دستگاه مجازی جاوا (JVM)، افزونه ی Gradle یک نسخه ی خاص از فایل android.jar را ارائه می نماید (این فایل تحت عنوان Android mockable jar نیز شناخته می شود). فایل نام برده در اختیار کل unit test قرار گرفته تا تمامی فیلد ها، متدها و کلاس ها در دسترس و قابل استفاده باشند. لازم به ذکر است که فراخوانی هر یک از توابع JAR، به صورت پیش فرض، منجر به صدور خطا (exception) می شود.

بنابراین اگر کلاس های شما توابع کتابخانه ای اندروید (Android API) را فراخوانی نمی کنند، شما می توانید از فریم ورک تست گیری JUnit (یا هر فریم ورک دیگری که برای تست اپلیکیشن های جاوا ارائه شده) بدون محدودیت استفاده نمایید. چنانچه به توابع کتابخانه ای اندروید (android API) وابستگی (dependency) دارید (مجبورید از کتابخانه ها و کلاس های API)

استفاده نمایید، در آن صورت این وابستگی ها در کد شما بایستی (برای unit test ها) از طریق فریم ورک های mocking (فریم ورک های ساختگی همچون Mockito) جایگزین و جبران گردد. برای مشاهده ای اطلاعات بیشتر جهت فعل سازی مقادیر بازگشتی برای متدهای ساختگی و شبیه سازی شده از فایل mockable.jar اندروید، به این آدرس مراجعه نمایید: Activating default .return values for mocked methods in android.jar

این قابلیت را در اختیار توسعه دهنده قرار می دهد تا تست های نرم افزاری برای اپلیکیشن های خود طراحی کند. کتابخانه‌ی مذکور AndroidJUnitRunner فریم ورک تست گیری Espresso و UI Automator را شامل می شود.

امکان ساخت و اجرای تست های JUnit 4 را فراهم می کند، در حالی که چارچوب تست گیری Espresso برای طراحی تست ویژه‌ی لایه‌ی UI نرم افزار مناسب می باشد. با استفاده از UI Automator نیز برنامه نویس قادر خواهد بود تست های cross functional برای اپلیکیشن اندرویدی طراحی نماید.

توابع کتابخانه‌ی AndroidJUnitRunner اندروید در قالب کلاس Instrumentation API (Instrumentation Registry) را در اختیار توسعه دهنده قرار می دهد.

- `InstrumentationRegistry.getInstrumentation()` تابعی است که در خروجی `Instrumentation` فعلی و در حال اجرا را برمی گرداند.
- `InstrumentationRegistry.getContext()` تابعی است که در خروجی `Context` پکیج `Instrumentation` جاری را بازگردانی می نماید.
- `InstrumentationRegistry.getTargetContext()` متدهایی است که اپلیکیشن `Context` را بازگردانی می نماید.
- `InstrumentationRegistry getArguments()` یک کپی از مجموعه `Bundle` آرگومان هایی که به این `Instrumentation` ارسال شده را در خروجی بازمی گرداند. این تابع زمانی

به کار شما می آید که بخواهید به آرگومان های command line ارسال شده به برای تست خود استفاده نمایید.

برای دسترسی و مدیریت lifecycle نیز می توانید از کلاس ActivityLifecycleMonitorRegistry استفاده نمایید.

## 11-7-ساختار پروژه اندرویدی و ایجاد پوشه تست

### 11-1-سازماندهی پروژه اندرویدی برای تست

برای سازمان دهی پروژه های تست اپلیکیشن، توسعه دهنده از قرار داد خاصی پیروی می کنند. جهت سازمان دهی کد اپلیکیشن، در این پروژه های اندرویدی توصیه می شود که از ساختار عنوان شده در زیر استفاده نمایید. این ساختار را ویزارد پروژه اندروید نیز به صورت جداگانه ارائه می دهد.

اگر از این قرارداد پیروی کنید، بی شک سیستم کامپایل اندروید (Android build system) قادر خواهد بود تست شما را به صورت خودکار بر روی دستگاه مورد نظر (JVM یا دستگاه واقعی اندروید) اجرا کند.

### 11-2-برطرف کردن خطای "error duplicate files in path"

در صورت برخورد با این خطای "error duplicate files in path. Path in archive: LICENSE.txt" ، می توانید با افزودن قطعه کد زیر به فایل app/gradle.build ، به راحتی خطای مذبور را برطرف نمایید.

```
android {
    packagingOptions {
        exclude 'LICENSE.txt'
    }
}
```

{}

## 12-7-اجرای Unit test بر روی JVM

### 12-7-اجرای Unit test بر روی نرم افزار در بستر Android runtime

برای تست هایی که (به صورت local) در محیط JVM یا در بستر دستگاه واقعی اندروید (Android runtime) اجرا می شوند، واژه unit tests را بکار می برد.

عبارت است از تستی که عملکرد و قابلیت یک کامپوننت نرم افزاری را به صورت جداگانه مورد آزمایش قرار می دهد.

اپلیکیشنی را در نظر بگیرید که در آن دکمه‌ای داخل یک activity، سبب فعال سازی و اجرای دیگری می شود. unit test ای که برای اپلیکیشن نوشته می شود، بررسی می کند آیا intent مورد نیاز (برای راه اندازی activity دوم) صادر می شود یا خیر. intent کاری با اینکه آیا intent دوم در نتیجه فراخوانده شدن intent اجرا می شود یا خیر، ندارد.

Unit test ها بر اساس ورژن ویرایش شده از کتابخانه ای اندروید android.jar اجرا می شوند. در این ویرایش تمامی modifier های final حذف شده اند. این امر به شما امکان می دهد تا از کتابخانه های شبیه سازی و ایجاد آبجکت های ساختگی (mocking library) همچون Mockito بهره بگیرید.

### 12-7- محل جایگذاری unit test ها در پروژه ای اندرویدی

همان طور که قبل ذکر شد، unit test های اندروید باید در پوشه app/src/test قرار داده شوند.

### 7-12-3-کتابخانه های الزامی در فایل dependency

برای استفاده و اجرای تست های JUnit بر روی کامپوننت های نرم افزاری مربوط به اپلیکیشن خود، بایستی آن ها را در قالب dependency (کتابخانه) به فایل Gradle build خود اضافه نمایید.

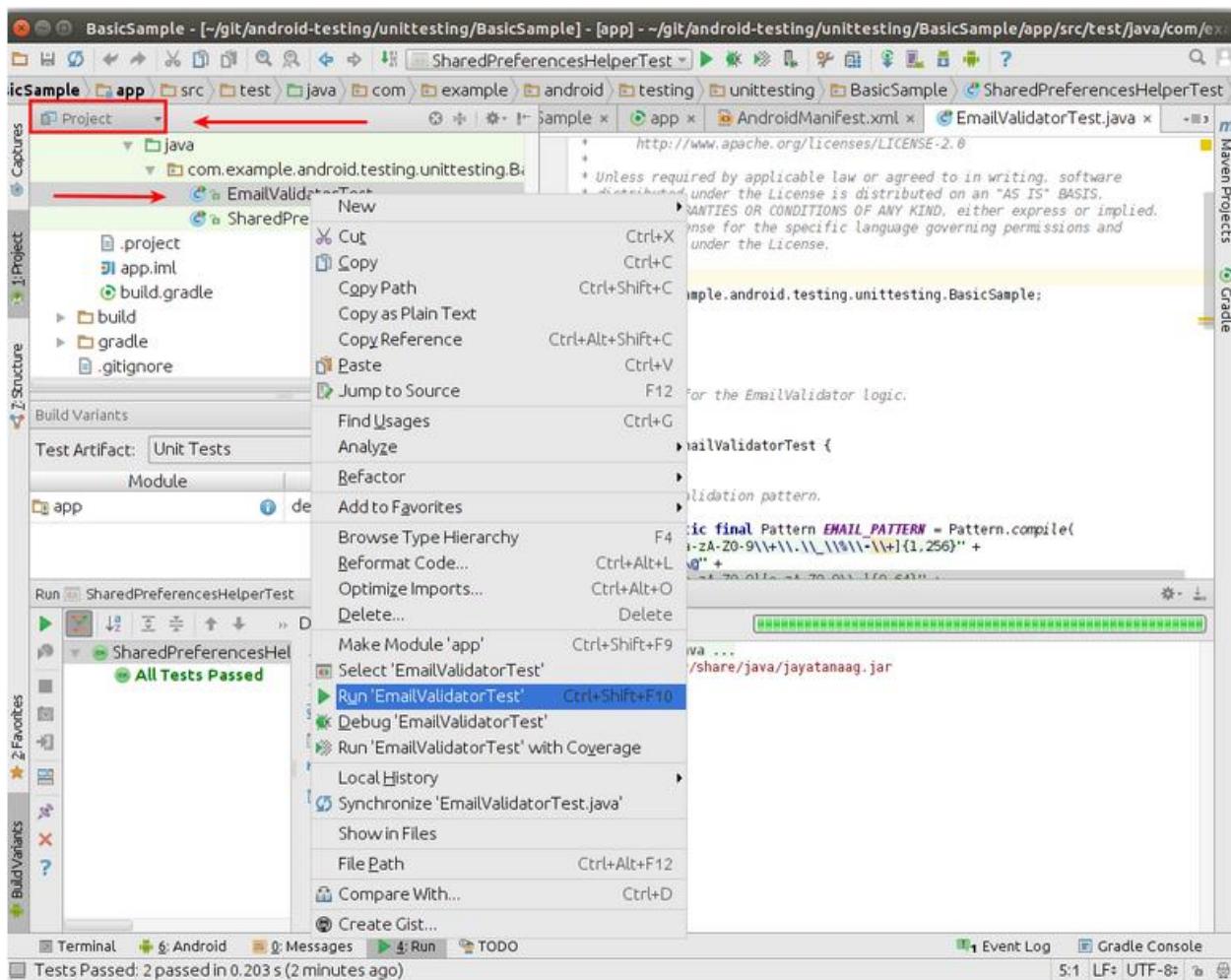
```
dependencies {
    // Unit testing dependencies
    testCompile 'junit:junit:4.12'
    // Set this dependency if you want to use the Hamcrest matcher library
    testCompile 'org.hamcrest:hamcrest-library:1.3'
    // more stuff, e.g., Mockito
}
```

### 7-12-4-اجرای unit test ها از طریق سیستم کامپایل Gradle

Unit test های خود را با فراخوانی دستور gradlew test اجرا نمایید.

### 7-12-5-اجرای unit test ها از محیط کاری Android Studio

به منظور اجرای یک unit test، بر روی کلاس تست گیری (test class) در پنجره ی Project در منوی Run راست کلیک کرده و سپس گزینه ی Run را انتخاب نمایید.



## ۱۲-۷- محل قرارگیری نتایج و گزارش های مربوط به تست (test reports)

گزارش ها و نتایج مربوط به تست در پوشه `app/build/reports/tests/debug/`directory` ایجاد می شوند. فایل `index.html` اطلاعاتی درباره هر یک از صفحات تست فراهم آورده و به آن ها جهت مشاهده اطلاعات بیشتر لینک می دهد.

## 7-12-7-فعال سازی مقادیر بازگشته پیش فرض متدهای ساختگی android.jar (mocked methods)

به طور پیش فرض، فراخوانی هر یک از توابع موجود در فایل android.jar سبب رخداد خطا (exception) می شود. این وضعیت پیش فرض باعث می شود که unit test ها تنها کد شما را تست کنند و به هیچ یک از رفتار خاص محیط یا بستر اجرای اندروید (Android platform) وابسته نباشند. در صورت نیاز به تنظیم اختصاصی یک قابلیت یا رفتار خاص، کافی است از فریم ورک های شبیه ساز (mocking framework) جهت جایگزینی این متدها و فراخوانی آن ها استفاده نمایید.

همچنین می توانید به سیستم کامپایل Gradle دستور بدید که به عنوان خروجی متدهای فراخوانی شده، مقادیر پیش فرض مورد نظر را بازگردانی نمایند. برای این منظور کافی است تنظیمات زیر را در فایل Gradle build اپلیکیشن خود لحاظ نمایید.

```
android {
    ...
    testOptions {
        unitTests.returnDefaultValues = true
    }
}
```

**تمرین: طراحی unit test جهت تست اپلیکیشن مرحله‌ی آماده سازی: ساخت پروژه‌ی اندروید**

پروژه‌ی اندرویدی خود را بر اساس برنامه‌ای که در مباحث قبلی ساختید (Android temperature converter)، طراحی نمایید.

**تمرین: ساخت unit test**

**هدف از انجام این تمرین**

طی تمرین حاضر قادر خواهید بود یک تست JUnit4 برای پروژه‌ی اندرویدی خود بنویسید.

## 8-7-افزودن dependency / کتابخانه‌ی JUnit

Dependency یا کتابخانه‌ی JUnit را در فایل app/build.gradle جایگذاری نمایید. اگر پوشه‌ی test در پروژه‌ی شما موجود نبود، در آن صورت بایستی طبق فرایندی که قبلاً توضیح داده شد (نحوه‌ی ساخت پوشه‌ی تست در محیط کاری (Android Studio) را طی نموده و پوشه‌ی مورد نیاز را ایجاد نمایید).

```
dependencies {
    // Unit testing dependencies
    testCompile 'junit:junit:4.12'
}
```

ساخت و طراحی تست

در پوشه‌ی app/src/test، دو متده‌ی تست گیری (test method) زیر را داخل کلاس ConverterUtil تعریف نمایید.

```
package com.vogella.android.temperature.test;
import static org.junit.Assert.*;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import com.vogella.android.temperature.ConverterUtil;
public class ConverterUtilTest {
    @Test
    public void testConvertFahrenheitToCelsius() {
        float actual = ConverterUtil.convertCelsiusToFahrenheit(100);
        // expected value is 212
        float expected = 212;
        // use this method because float is not precise
        assertEquals("Conversion from celsius to fahrenheit failed", expected,
                     actual, 0.001);
    }
    @Test
    public void testConvertCelsiusToFahrenheit() {
        float actual = ConverterUtil.convertFahrenheitToCelsius(212);
        // expected value is 100
        float expected = 100;
    }
}
```

```

    // use this method because float is not precise
    assertEquals("Conversion from celsius to fahrenheit failed", expected,
        actual, 0.001);
}
}
}

```

## 7-12-9- اجرای unit test بر روی پروژه

بد نیست با تست unit هایی که برای بخش های مختلف پروژه‌ی خود نوشته اید، اطمینان حاصل نمایید که تست‌ها به درستی پیاده سازی شده‌اند. تست‌های شما بایستی طبق انتظار به درستی اجرا شوند.



## 7-13- طراحی instrumentation test برای اجرای تست بر روی اپلیکیشن در بستر دستگاه حقیقی اندروید

Instrumentation test-7-13-1  
API یا توابع کتابخانه‌ای تست گیری اندروید hook هایی را برای وصل شدن به کامپوننت‌های نرم افزاری و lifecycle اپلیکیشن اندروید فراهم می‌آورند (hook) مکان یا interface ای است که در قالب کد پکیج شده ارائه شده و به برنامه نویس اجازه می‌دهد تا کدهایی با تنظیمات اختصاصی و دلخواه وارد متن برنامه کند. به طور مثال، توسعه دهنده می‌تواند کدی وارد برنامه کند که وظیفه‌ی آن بررسی این است که هر چند وقت یکبار یک مسیر منطقی داخل اپلیکیشن مورد نظر طی می‌شود).

این hook ها در اصطلاح instrumentation API خوانده شده و به تست های شما این اجازه را می دهد تا اداره‌ی lifecycle و event های مرتبط با تعامل کاربر را بدست بگیرند.

در شرایط معمولی، اپلیکیشن نمی تواند event های مربوط به lifecycle را کنترل کند و در این برهه کاربر مدیریت جریان یا روند اپلیکیشن را بدست می گیرد. برای مثال، زمانی که سیستم اندروید کامپوننت activity اپلیکیشن را اجرا می کند، در واقع متدهای onCreate() فراخوانده می شود. یا هنگامی که کاربر دکمه ای را در نمایشگر فشار می دهد، کد مربوطه از اپلیکیشن فراخوانده و اجرا می شود. توسعه دهنده می تواند با بهره گیری از توابع کتابخانه ای instrumentation، رخدادهایی از این دست را با اجرای کد آزمایشی (test code) خود اداره کند. سپس می تواند متدهای finish() را صدا زده، activity را مجدد اجرا کند و بدین وسیله مطمئن شود که اطلاعات مربوط به وضعیت activity (instance state) مذبور پس از راه اندازی مجدد بازگردانی می شوند یا خیر.

Unit test هایی که زیرمجموعه‌ی instrumentation هستند، همان طور که پیش‌تر نیز ذکر شد، بجای JVM (دستگاه مجازی جاوا)، بر روی دستگاه‌های حقیقی اندروید یا در بستر نرم افزار شبیه‌ساز (emulator) اجرا می شوند. این تست‌ها به دستگاه واقعی اندروید و منابع آن دسترسی داشته و برای اجرای تست بر روی کامپوننت‌های مجازی نرم افزار و انجام unit test بسیار ساده می باشند. همان طور که می دانید، تست قابلیت‌های اپلیکیشن به واسطه‌ی unit test با بهره گیری از فریم‌ورک‌های شبیه‌سازی (mocking frameworks) به راحتی امکان پذیر نمی باشد. به عنوان مثال می توان به تستی اشاره کرد که داده‌سازی یک Parcelable را آزمایش کرده و سعی می کند از عملکرد صحیح آن اطمینان حاصل نماید.

کلاس تست گیری که مبتنی بر instrumentation می باشد، این امکان را به شما می دهد تا event های اصلی (event های مربوط به نمایشگر/touch event) را به اپلیکیشن مورد تست ارسال نمایید.

با وجود فریم‌ورک‌های تست گیری همچون Espresso که قابلیت آزمایش UI را فراهم می آورند، توسعه دهنده به ندرت مجبور می شود تا مستقیماً از instrumentation API استفاده نماید.

## 7-13-2- سیستم اندروید چگونه تست ها را اجرا می کند

(کلاس تست گیری) پایه برای اجرای تست های نرم افزاری بر روی اپلیکیشن های اندرویدی، `InstrumentationTestRunner` است. این کلاس تمامی متدهای تست گیری را اجرا کرده و در حافظه بارگذاری می نماید. سپس به واسطه ای `instrumentation API` با سیستم اندروید ارتباط برقرار می کند. زمانی که تستی را جهت آزمایش اپلیکیشن اندرویدی خود اجرا می کنید، سیستم اندروید تمامی فرایندهای مربوط اپلیکیشن مورد آزمایش (که در حال تست شدن هستند) را از حافظه پاک کرده و متعاقباً یک نمونه (`instance`) جدید از آن را در حافظه بارگذاری می کند. لازم به توضیح است که کلاس مزبور اپلیکیشن را راه اندازی نمی کند، بلکه این امر، وظیفه ای متدهای تست گیری می باشد. به عبارت دیگر این متدهای تست گیری است که اداره `lifecycle` کامپوننت های نرم افزاری را بدست می گیرد.

کلاس تست گیری مورد نظر، به هنگام راه اندازی و مقداردهی اولیه (`initialization`، همچنین متدهای `onCreate()` مربوط به اپلیکیشن و `activity` مورد تست را صدا می زند.

## 7-13-3- شبیه سازی رفتار آبجکت ها در اندروید (ایجاد آبجکت های ساختگی (جهت تست)

توسعه دهنده برای اجرای تست های `instrumentation` بر روی نرم افزار اندرویدی خود، همچنین می تواند از فریم ورک شبیه سازی (mocking framework) `Mockito` استفاده نماید. با بهره گیری از این فریم ورک توسعه قادر خواهد بود آن بخش هایی از سیستم اندروید را که تست آن ها ضروری ندارد، با بخش هایی از این فریم ورک جایگزین نماید. ویرایش های قبلی چارچوب نرم افزاری اندروید (`Android framework`) کلاس های شبیه سازی / `mocking classes` را ویژه تست بخش های اپلیکیشن ارائه می دادند. اما امروزه با وجود `Mockito` این کلاس ها دیگر کاربردی ندارند.

## 7-13-4- محل قرارگیری تست های instrumentation

همان طور که قبل توضیح داده شد (در بخش سازماندهی پروژه و تست ها)، unit test ها بایستی در پوشه‌ی `app/src/androidTest/java` قرار گیرد.

## 7-13-5- تعریف testInstrumentationRunner ها و dependency داخل Gradle build

جهت استفاده از JUnit برای اپلیکیشن های اندرویدی خود، می‌بایست dependency (کتابخانه) مورد نیاز را به فایل Gradle build اضافه نمایید. علاوه بر آن لازم است `android.support.test.runner.AndroidJUnitRunner` را نیز در قالب کلاس `testInstrumentationRunner` داخل فایل build تعریف نمایید.

```
defaultConfig {
    .... more stuff
    testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
}
dependencies {
    // Unit testing dependencies
    androidTestCompile 'junit:junit:4.12'
    // Set this dependency if you want to use the Hamcrest matcher library
    androidTestCompile 'org.hamcrest:hamcrest-library:1.3'
    // more stuff, e.g., Mockito
}
```

## 7-13-6- استفاده از @RunWith(AndroidJUnit4.class)

توصیه می‌شود تست را با دستور `@RunWith(AndroidJUnit4.class)` نیز نشانه گذاری نمایید. `AndroidJUnit4` از `JUnit4` ارث بری می‌کند (annotate) از syntax/ساختار نگارشی خالص `junit4` و `ActivityTestRule` استفاده نمایید، دیگر لزومی ندارد تست را با annotation فوق نشانه گذاری نمایید. اما در صورتی که لازم باشد تست های فریم ورک Espresso را همراه با `ActivityTestRule` و `JUnit4` اجرا نمایید، استفاده از آن الزامی خواهد بود.

## 7-13-7- اجرای unit test از طریق سیستم کامپایل Gradle

Unit test را با فراخوانی دستور gradlew connectedCheck اجرا نمایید.

## 7-13-8- اجرای unit test از داخل محیط برنامه نویسی Android Studio

بر روی کلاس تست گیری در پنجره Project راست کلیک کرده و سپس گزینه Run را انتخاب نمایید.

## 7-13-9- محل جایگذاری گزارش های تست

گزارش های تست در پوشه app/build/reports/androidTests/connected/ می شود.

فایل index.html اطلاعات مختصری درباره تست ارائه داده و به تمامی صفحات آن لینک می دهدند.

## 7-13-10- نحوه جایگزین کردن بخش های اپلیکیشن با instrumentation test ها

می توانید برای اجرای تست های مبتنی بر instrumentation و جایگزین کردن کلاس اپلیکیشن (با تست های newApplication و AndroidJUnitRunner) کلاس (InstrumentationTest) را بازنویسی override نمایید.

```
package com.vogella.android.daggerjunitmockito;
import android.app.Application;
public class MyMockApplication extends Application {
    @Override
    public void onCreate() {
        // do something important for your tests here
    }
}
```

:Test runner

```
package com.vogella.android.daggerjunitmockito;
import android.app.Application;
import android.content.Context;
import android.support.test.runner.AndroidJUnitRunner;
public class MockTestRunner extends AndroidJUnitRunner {
    @Override
    public Application newApplication(ClassLoader cl, String className, Context context)
        throws InstantiationException, IllegalAccessException, ClassNotFoundException {
        return super.newApplication(cl, MyMockApplication.class.getName(), context);
    }
}
```

همچنین لازم است این test runner را در فایل build.gradle ثبت (تعریف) نمایید.

```
android {
    // more
    testInstrumentationRunner "com.vogella.android.daggerjunitmockito.MockTestRunner"
}
// more
}
```

**تمرین: شبیه سازی و کسب اطمینان از عملکرد صحیح قابلیت دسترسی به فایل (Mocking file access)**

**هدف از این تمرین**

لازم نیست این کلاس را در اپلیکیشن اندرویدی خود بکار ببرید و در اینجا تنها به منظور نمایش نحوه استفاده از Mockito در unit test Mocking file access شده است.

### 11-13-7- ایجاد کلاسی جهت تست گیری

در اپلیکیشن اندرویدی آماده ای خود و یا پروژه ای با پکیج `.com.vogella.android.testing.mockitocontextmock` کلاس زیر را پیاده سازی نمایید.

```
public class Util {
    public static void writeConfiguration(Context ctx ) {
        try (FileOutputStream open FileOutputStream =
            ctx.openFileOutput( "config.txt", Context.MODE_PRIVATE);) {
            open FileOutputStream.write("This is a test1.".getBytes());
            open FileOutputStream.write("This is a test2.".getBytes());
        } catch (Exception e) {
```

```
        // not handled  
    }  
}  
}
```

ساخت unit test- ۱۳-۱۲- جدید

با استفاده از فریم ورک شبیه سازی (mocking framework)، یک unit test جدید بنویسید که موارد زیر را بررسی می کند:

openFileOutput دقتاً یکبار فراخوانی شود.

```
package com.vogella.android.testing.mockitocontextmock;
import android.content.Context;
import org.junit.Before;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.mockito.Mockito;
import org.mockito.MockitoAnnotations;
import java.io.FileOutputStream;
import static org.junit.Assert.fail;
import static org.mockito.Matchers.any;
import static org.mockito.Matchers.anyInt;
import static org.mockito.Matchers.anyString;
import static org.mockito.Mockito.atLeast;
import static org.mockito.Mockito.Mockito.times;
import static org.mockito.Mockito.verify;
import static org.mockito.Mockito.Mockito.when;
public class TextContextOutputStream {
    @Mock
    Context context;
    @Mock
    FileOutputStream fileOutputStream;
    @Before
    public void init(){
        MockitoAnnotations.initMocks(this);
    }
    @Test
    public void writeShouldWriteTwiceToFileSystem() {
        try {
            when(context.openFileOutput(anyString(), anyInt())).thenReturn(fileOutputStream);
            Util.writeConfiguration(context);
        } catch (IOException e) {
            fail("Exception while writing configuration");
        }
    }
}
```

```

        verify(context, times(1)).openFileOutput(anyString(), anyInt());
        verify(fileOutputStream, atLeast(2)).write(any(byte[].class));
    } catch (Exception e) {
        e.printStackTrace();
        fail();
    }
}
}
}

```

## 7-14-1-اطلاعات بیشتر در خصوص اجرای تست بر روی اپلیکیشن های اندرویدی

### 7-14-1-assert کلاس های

توابع کتابخانه ای اجرای تست (Android testing API) بر روی اپلیکیشن های اندرویدی، علاوه بر JUnit Assert، دو کلاس دیگر به نام های MoreAsserts و ViewAsserts ارائه می دهد.

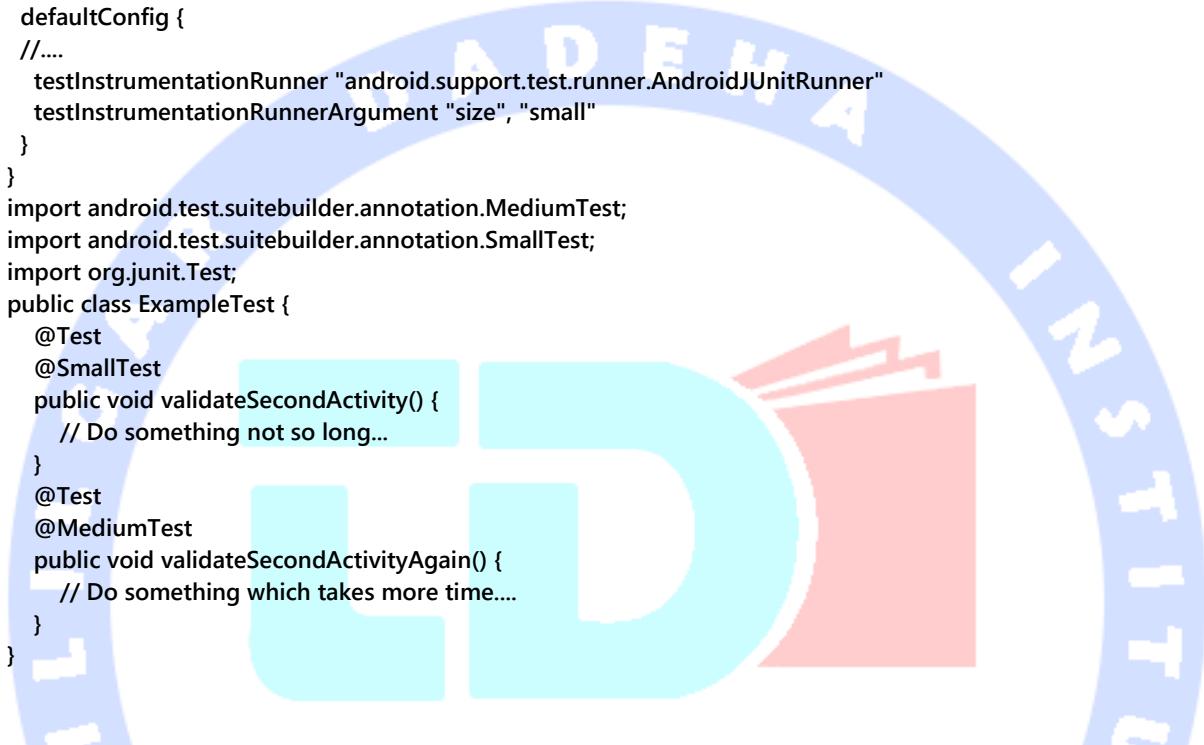
### 7-14-2-Test group ها (گروه بندی تست ها)

دستورات `@LargeTest`، `@MediumTest` و `@SmallTest` annotations امکان گروه بندی تست ها را برای توسعه دهنده فراهم می کنند. به عبارتی روشن تر این قابلیت توسعه دهنده را قادر می سازد تا تست ها را بر اساس ویژگی خاصی گروه بندی نماید. برای مثال تنها آن دسته از تست هایی را بر روی نرم افزار اجرا کند که زمان اجرای آن ها چندان طولانی نیست یا تست هایی که زمان اجرای آن ها طولانی هستند را بر روی سرویس دهنده *continuous integration* به اجرا بگذارند (*Continuous integration* به یک سری روش گفته می شود که طی آن سلامت نرم افزار به طور دائم تضمین می شود، به طوری که خورده تغییرات افراد درگیر در پروژه باعث رخداد خطأ در کل پروژه نشده و به درستی مثل تکه های پازل در کنار هم قرار گیرند).

برای اینکه تنها تست های انتخابی اجرا شوند، شما می توانید کلاس *InstrumentationTestRunner* را به واسطه ای افزونه *(plug-in)* Gradle تنظیم نمایید. کد

زیر نمونه ای از فایل build.gradle را نمایش می دهد که با دستور @SmallTests علامت گذاری شده و به همین خاطر تنها تست هایی که زمان اجرای طولانی ندارند را اجرا می کند.

```
android {
//...
defaultConfig {
//...
    testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    testInstrumentationRunnerArgument "size", "small"
}
}
import android.test.suitebuilder.annotation.MediumTest;
import android.test.suitebuilder.annotation.SmallTest;
import org.junit.Test;
public class ExampleTest {
    @Test
    @SmallTest
    public void validateSecondActivity() {
        // Do something not so long...
    }
    @Test
    @MediumTest
    public void validateSecondActivityAgain() {
        // Do something which takes more time....
    }
}
```



### 3-14-7- فیلتر کردن تست

می توانید تست های نرم افزاری خود را با annotation ها نشانه گذاری کرده و از این طریق تست های مزبور را فیلتر نمایید (برای مثال شرایطی که تست در آن اجرا می شود را دقیق مشخص کنید). در واقع کلاس اجرای تست های نرم افزاری اندروید (android test runner) به شما امکان می دهد این تست ها را به وسیله annotation های زیر، محدود ساخته و صریحا مشخص کنید که تست با چه شرایطی بایستی اجرا شود.

Annotation های فیلتر و محدود سازی تست	
Annotation	شرح کاربرد
@RequiresDevice	تعیین می نماید که تست تنها بایستی بر روی دستگاه های واقعی اندروید اجرا شود. چنانچه محیط اجرای تست شبیه ساز باشد، تست به واسطه ای این annotation اجرا نخواهد شد.

@SdkSupress

@SDKSupress(minSdkVersion=18)

## 7-14-4- دستور @FlakyTest

می توانید با استفاده از دستور `@FlakyTest` به سیستم اندروید اعلان نمایید که تست را در صورت عدم موفقیت، بار دیگر اجرا کند. Annotation ذکر شده، یک attribute به نام tolerance دارد که با مقدار دهی آن می توانید تعیین نمایید که تست پس از چندبار ناموفق بودن در اجرا، failed در نظر گرفته شده و دیگر تکرار نشود.

### تمرین: تست lifecycle

#### هدف از این تمرین

در این تمرین، استفاده از فریم ورک تست گیری نرم افزار در بستر دستگاه حقيقی اندروید (instrumentation test) به صورت عملی نمایش داده شده است. البته برای چنین تستی، بهتر است از فریم ورک های سطح بالاتر همچون Espresso استفاده نمایید.

### ایجاد پروژه و تست آن

یک پروژه و activity جدید به ترتیب به نام های `com.vogella.android.test.simpleactivity` و `MainActivity` تعریف نمایید.

حال activity دیگری به نام `SecondActivity` تعریف نموده و به پروژه ای نام بردۀ اضافه نمایید. این activity بایستی از یک layout استفاده نموده و حداقل یک آبجکت `TextView` دربرداشته باشد. id این آبجکت بایستی بر روی "resultText" تنظیم شده و آن با "Started" مقداردهی شده باشد.

یک فیلد `EditText` به فایل `MainActivity` کلاس layout اضافه نمایید.

یک آبجکت دکمه به فایل layout ای که مورد استفاده ای کلاس MainActivity می باشد، اضافه نمایید. زمانی که دکمه کلیک می شود، دومین activity بایستی اجرا شود.

فیلد متنی EditText را با استفاده از "text" به عنوان کلید آن، داخل آبجکت intent قرار دهید. پس از آن رشته ای http://www.vogella.com را به عنوان extra (اطلاعات اضافی که بین دو activity مبادله می شود) با انتخاب "URL" به عنوان کلید، (به وسیله ای دستور intent.putExtra("URL", "http://www.vogella.com"); در زیر نمونه کدی از پیاده سازی MainActivity را مشاهده می کنید.

```
package testing.android.vogella.com.simpleactivity;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void onClick(View view) {
        Intent intent = new Intent(this, SecondActivity.class);
        intent.putExtra("URL", "http://www.vogella.com");
        startActivity(intent);
    }
}
```

### 7-14-5 ساخت کلاس و پروژه ای تست

یک instrumentation test دیگر مشابه زیر پیاده سازی نمایید.

```
package com.vogella.android.test.simpleactivity.test;
import android.app.Activity;
import android.app.Instrumentation;
import android.app.Instrumentation.ActivityMonitor;
import android.test.ActivityInstrumentationTestCase2;
import android.test.TouchUtils;
import android.test.UiThreadTest;
import android.test.ViewAsserts;
```

```

import android.view.KeyEvent;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import com.vogella.android.test.simpleactivity.R;
import com.vogella.android.test.simpleactivity.MainActivity;
import com.vogella.android.test.simpleactivity.SecondActivity;
public class SecondActivityFunctionalTest extends
    ActivityInstrumentationTestCase2<SecondActivity> {
private static final String NEW_TEXT = "new text";
public SecondActivityFunctionalTest() {
    super(SecondActivity.class);
}
public void testSetText() throws Exception {
    SecondActivity activity = getActivity();
    // search for the textView
    final TextView textView = (TextView) activity
        .findViewById(R.id.resultText);
    // set text
    getActivity().runOnUiThread(new Runnable() {
        @Override
        public void run() {
            textView.setText(NEW_TEXT);
        }
    });
    getInstrumentation().waitForIdleSync();
    assertEquals("Text incorrect", NEW_TEXT, textView.getText().toString());
}
@UiThreadTest
public void testSetTextWithAnnotation() throws Exception {
    SecondActivity activity = getActivity();
    // search for the textView
    final TextView textView = (TextView) activity
        .findViewById(R.id.resultText);
    textView.setText(NEW_TEXT);
    assertEquals("Text incorrect", NEW_TEXT, textView.getText().toString());
}
}

```

## 7-15-استفاده از ابزار Monkey جهت ایجاد و ارسال شبه event به دستگاه

### 7-15-1-توضیحی درباره ابزار monkey

یک ابزار command-line است که بر روی محیط شبیه ساز یا دستگاه حقیقی اندروید اجرا شده و این قابلیت را به شما می دهد تا علاوه بر مجموعه ای از event هایی که توسط کاربر فرخوانی می شوند، تعدادی از event هایی که در سطح سیستم رخ می دهند را نیز به صورت

شبیه تصادفی ایجاد و به سیستم ارسال نمایید. در واقع شما با این ابزار قادر خواهید بود اپلیکیشن های در حال توسعه‌ی خود را به صورت تصادفی اما تکرار پذیر، تحت شرایط سنگین آزمایش نموده و مطمئن شوید که اپلیکیشن مورد نظر در صورت وجود بار کاری زیاد نیز دارای قابلیت دسترسی و مدیریت خطای بالایی می‌باشد (stress-test).

شما می‌توانید ابزار مزبور را طوری تنظیم کنید که تنها برای پکیج خاصی اجرا شده و صرفاً اپلیکیشن مد نظر را تست کند.

## 2-15-7- نحوه‌ی استفاده از ابزار Monkey

دستور زیر 2000 رخداد تصادفی به اپلیکیشنی که اسم پکیج de.vogella.android.test.target است را در بالای فایل خود دارند، ارسال می‌کند.

```
adb shell monkey -p de.vogella.android.test.target -v 2000
```

لازم به ذکر است که Monkey در برخی شرایط مشکلاتی را برای سرویس دهنده‌ی adb ایجاد می‌کند. در چنین موقعیتی می‌توانید با اجرای دو دستور زیر، سرویس دهنده‌ی adb را مجدداً راه اندازی نموده و مشکل را برطرف کنید.

```
adb kill-server  
adb start-server
```

برای اینکه مطمئن شوید توالی یا مجموعه event‌های تولید شده همیشه یکسان می‌باشد، باید پارامتر [seed] -s را به دستور اضافه نمایید.

## 7-16- تست آبجکت Application

در اندروید، منطق اپلیکیشن، داده‌ها و تنظیمات مربوط به کل برنامه داخل آبجکتی به نام application جای گرفته است. لازم است این آبجکت را تست کرده و از عملکرد صحیح آن اطمینان حاصل نمایید.

می توانید یک تست ۴ JUnit برای آزمایش عملکرد اپلیکیشن نوشته و آن را بر روی JVM اجرا نمایید. در این سناریو بایستی تمامی dependency و کتابخانه‌ی های مربوط به آبجکت اپلیکیشن را شبیه سازی (mock) نمایید.

برای تست آبجکت application در runtime اندروید، می بایست از کلاس ApplicationTestCase استفاده نمایید. در آینده انتظار می رود که شرکت گوگل یک rule جدید برای تست آبجکت یاد شده، در کتابخانه‌ی JUnit4 تعییه نماید. اما در زمان حاضر چنین امکانی برای استفاده‌ی توسعه دهنده فراهم نیست.

/ کلاس اجرای تست محیط (InstrumentationTestRunner) Android / test runner مقداردهی اولیه، به صورت خودکار یک نمونه از کلاس application می سازد. بنابراین در صورت استفاده از پردازش ناهمزمان در بدنه‌ی متده (onCreate)، لازم است حتماً این نکته را به خاطر داشته باشید.

## تمرین: تست اپلیکیشن ایجاد پروژه

یک اپلیکیشن جدید اندروید با اسم پکیج com.vogella.android.testing.applicationtest بر اساس قالب آماده‌ی (template) Empty Activity ایجاد نمایید. آبجکت application را با پیاده سازی زیر به برنامه‌ی خود اضافه نمایید.

```
package com.vogella.android.testing.applicationtest;
import android.app.Application;
import java.util.ArrayList;
import java.util.List;
public class MyApplication extends Application {
    public static final List<String> list = new ArrayList<String>();
}
```

لازم است اپلیکیشن را در لایه‌ی XML، داخل فایل manifest نیز اعلام نمایید.

```
<?xml version="1.0" encoding="utf-8"?>
```

```

<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.vogella.android.testing.applicationtest" >
    <application
        android:name=".MyApplication"
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>

```

## 16-7- تعریف unit test برای تست آبجکت application

داخل پوشه‌ی app/src/test یک unit test جدید تعریف نمایید. تست را طوری بنویسید که انتظار داشته باشد (assert) فیلد list (size) اولیه‌ی آن 0 باشد.

## 16-2- طراحی آبجکت instrumentation test برای application

یک unit test جدید بر اساس فریم‌ورک تست گیری 3 Unit به صورت زیر تعییه نمایید.

```

package com.vogella.android.testing.applicationtest;
import android.content.pm.PackageInfo;
import android.test.ApplicationTestCase;
import android.test.MoreAsserts;
public class ApplicationTest extends ApplicationTestCase<MyApplication> {
    private MyApplication application;
    public ApplicationTest() {
        super(MyApplication.class);
    }
    protected void setUp() throws Exception {
        super.setUp();
        createApplication();
        application = getApplication();
    }
    public void testCorrectVersion() throws Exception {
        PackageInfo info = application.getPackageManager().getPackageInfo(application.getPackageName(), 0);
        assertNotNull(info);
        MoreAsserts.assertMatchesRegex("\\d\\.\\d", info.versionName);
    }
}

```

## 17-7- تست سایر کامپوننت های نرم افزاری اندروید

### 17-7-1- تست سرویس

جهت تست یک سرویس، بایستی از کلاس ServiceTestRule که از Testing Support Library استفاده نمایید. در ویرایش های قبلی به منظور آزمایش کامپوننت سرویس از ServiceTestCase استفاده می شد که با ارائه ی کلاس نام برده، استفاده از آن منسوب شد.

این rule یک مکانیزم ساده در اختیار توسعه دهنده قرار می دهد که سرویس را قبل از اجرای تست راه اندازی نموده و با اتمام آن، کامپوننت مورد نظر را متوقف می سازد. در واقع این rule تضمین می کند که سرویس به هنگام اجرا به طور کامل متصل شده باشد. می توانید سرویس را به واسطه ی یکی از توابع کمکی (helper method) فراخوانی نمایید. زمانی که تست پایان می یابد، سرویس به صورت خودکار متوقف شده و به همراه آن تمامی متدهایی که با `@After` نشانه گذاری شده اند نیز خاتمه می یابند.

**نکته:** این rule همراه با IntentService قابل استفاده نمی باشد چرا که به مجرد اتمام اجرای متدهای `onHandleIntent`، به طور خودکار از حافظه پاک می گردد.

کد زیر یک سرویس را تست می کند.

```
@RunWith(AndroidJUnit4.class)
@MediumTest
public class MyServiceTest {
    @Rule
    public final ServiceTestRule mServiceRule = new ServiceTestRule();
    // test for a service which is started with startService
    @Test
    public void testWithStartedService() {
        mServiceRule.startService(new Intent(InstrumentationRegistry.getTargetContext(),
                MyService.class));
        // test code
    }
    @Test
    // test for a service which is started with bindService
    public void testWithBoundService() {
```

```

IBinder binder = mServiceRule.  

    bindService(new Intent(InstrumentationRegistry.getTargetContext(),  

        MyService.class));  

MyService service = ((MyService.LocalBinder) binder).getService();  

assertTrue("True wasn't returned", service.doSomethingToReturnTrue());
}
}

```

adb kill-server  
adb start-server

## 7-17-2- تست کامپوننت نرم افزاری Content provider

به منظور تست content provider، توسعه دهنده می تواند از کلاس ProviderTestCase2 استفاده کند. این کلاس به صورت خودکار از کامپوننت (content provider) مورد تست، نمونه سازی کرده و آبجکت IsolatedContext را در آن درج می نماید. گرچه این context یا بستر از سیستم اندروید مجزا است، اما همچنان اجازه ی دسترسی به فایل و دیتابیس را فراهم می کند. در واقع استفاده از آبجکت IsolatedContext این اطمینان را به وجود می آورد که provider مورد تست، تاثیری بر روی دستگاه واقعی اندروید نمی گذارد.

کلاس نام بردۀ همچنین با ارائه ی متodi به نام getMockContentResolver() امکان دسترسی به کلاس MockContentResolver را فراهم می سازد.

شما با استفاده تمامی عملیات مربوط به provider را آزموده و همچنین بررسی نمایید که در صورت فراخوانی provider با URI یا projection غیرمجاز چه اتفاقی رخ می دهد.

## 7-17-3- Loader تست

جهت تست کارکرد یک loader، توسعه دهنده می بایست از کلاس `LoaderTestCase` استفاده نماید. البته در آینده ی نزدیک انتظار می رود که شرکت گوگل با معرفی یک rule جدید در کتابخانه JUnit4، کلاس نام بردۀ را منسوخ نماید.

## Annotation-18-7 های متفرقه یا پشتیبان

علاوه بر annotation های نام برد، شرکت گوگل یک پکیج حاوی annotation های جدید در اختیار توسعه دهنده قرار می دهد. پکیج مزبور تعدادی metadata annotation (دستوراتی که اطلاعاتی را درباره ی عملکرد کد شرح می دهند) ارائه می دهد که با استفاده از آن ها در متن برنامه، توسعه دهنده قادر است از رخداد خطأ در عملکرد اپلیکیشن جلوگیری نماید.

جهت استفاده از این annotation ها، کافی است کتابخانه ی (dependency) زیر را به فایل Gradle build خود اضافه نماید.

```
dependencies {
    compile 'com.android.support:support-annotations:22.2.0'
}
```

جهت دریافت اطلاعات بیشتر درباره ی annotation ها می توانید به آدرس <http://tools.android.com/tech-docs/support-annotations> مراجعه نمایید.

## Annotation-7-18-1 Nullness های مربوط به

با درج دستور @Nullable در بالای کد توسعه دهنده به سیستم اعلان می کند که یک پارامتر یا مقدار بازگشتی می تواند null باشد. به طور مشابه، دستور @NonNull در بالای کد مربوطه نشانگر این است که یک پارامتر (یا مقدار بازگشتی) نمی تواند و نباید null باشد.

## Annotation-7-18-2 thread های مربوطه به

در صورتی که یک متدها از thread خاصی قابل فراخوانی باشد، می توانید آن را با یکی از annotation های زیر نشانه گذاری نمایید.

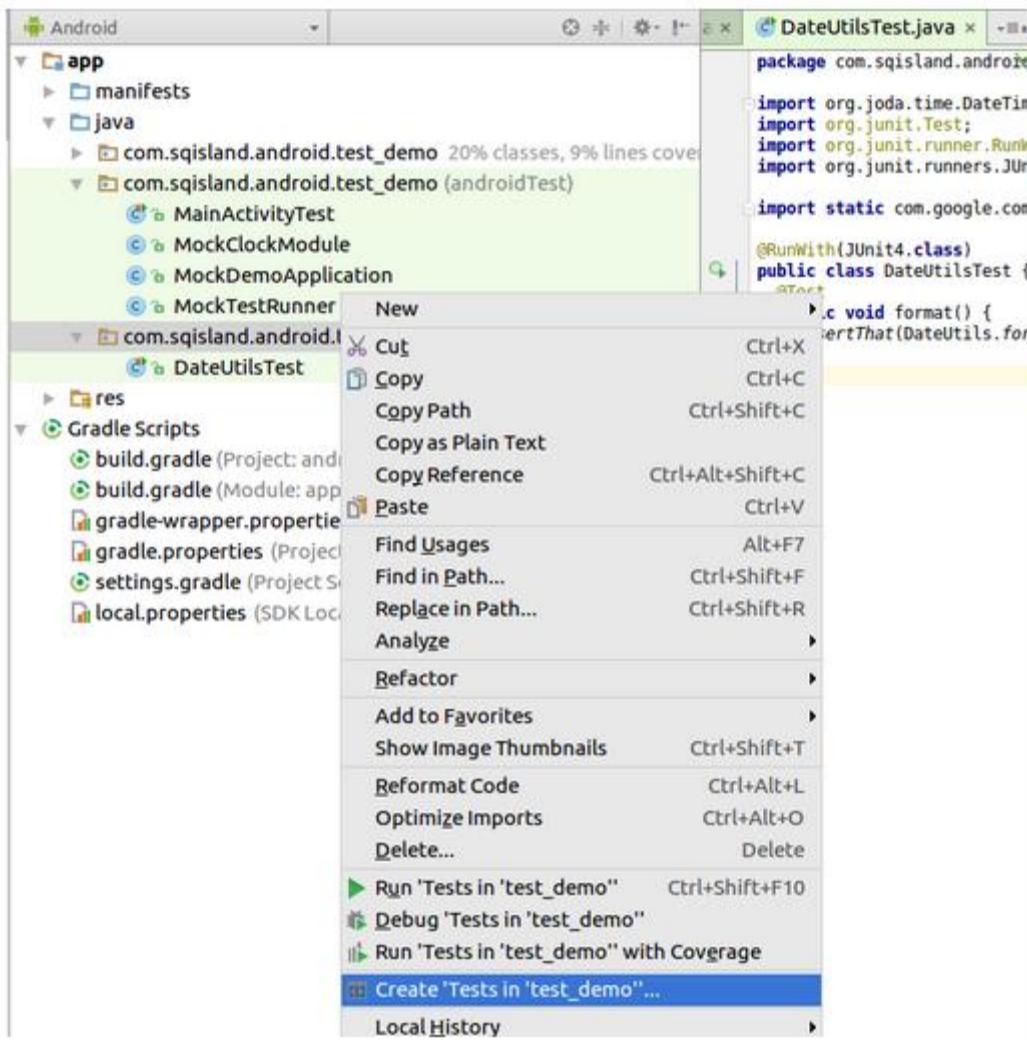
- `@UiThread`
- `@MainThread`
- `@WorkerThread`
- `@BinderThread`

مثال:

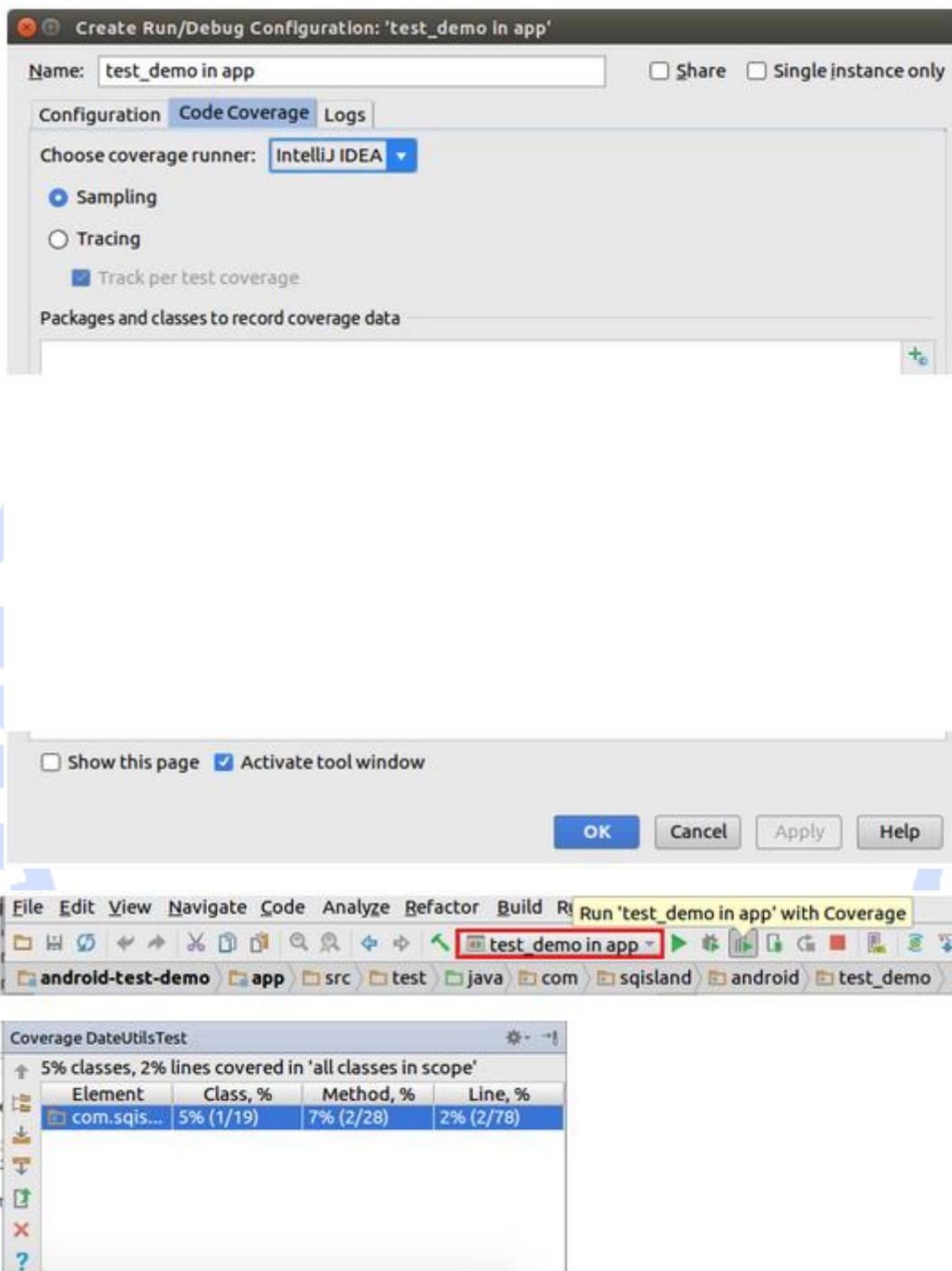
```
@WorkerThread
protected abstract Result doInBackground(Params... params);
@MainThread
protected void onProgressUpdate(Progress... values) {
}
```

### 7-18-3- تهیه‌ی گزارش از مقدار کد تست شده از کل پروژه (Code coverage report)

بیانگر این است چه میزان از کل کد اپلیکیشن شما توسط unit test مورد آزمایش قرار گرفته است. به منظور تهیه‌ی چنین گزارشی، می‌توانید تنظیمات جداگانه‌ای برای اجرا (launch config) آن تعییه نمایید. برای نیل به این هدف، پکیج مورد نظر خود را انتخاب نموده و پس از باز نمودن منو، آیتم [Create Tests in...] را انتخاب نمایید.

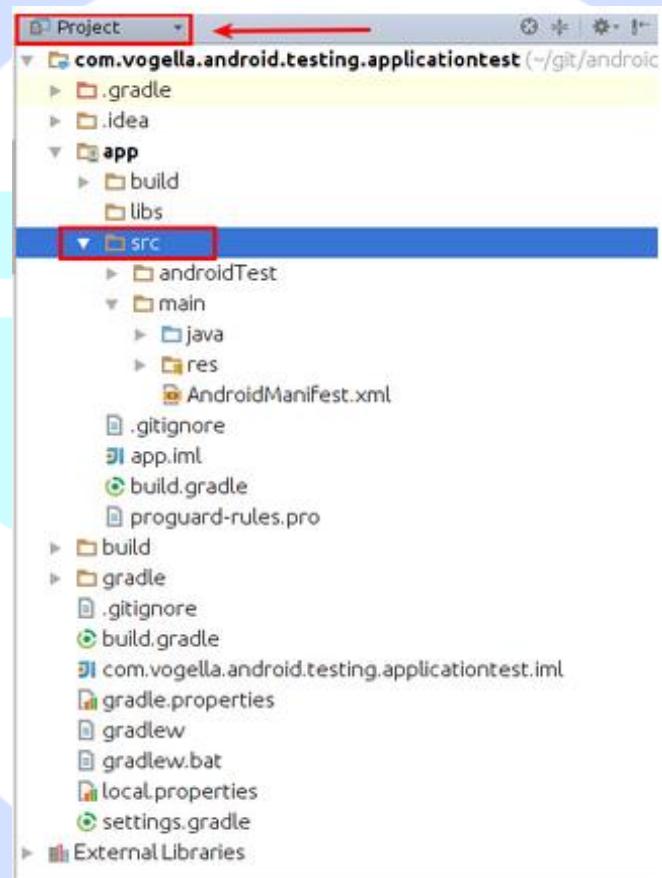


حال می توانید تنظیمات (config) جدید برای code coverage runtime (config) یا میزان کد مورد آزمایش خود ایجاد نمایید. پس از اجرا، یک گزارش تهیه شده و به نمایش گذاشته می شود.

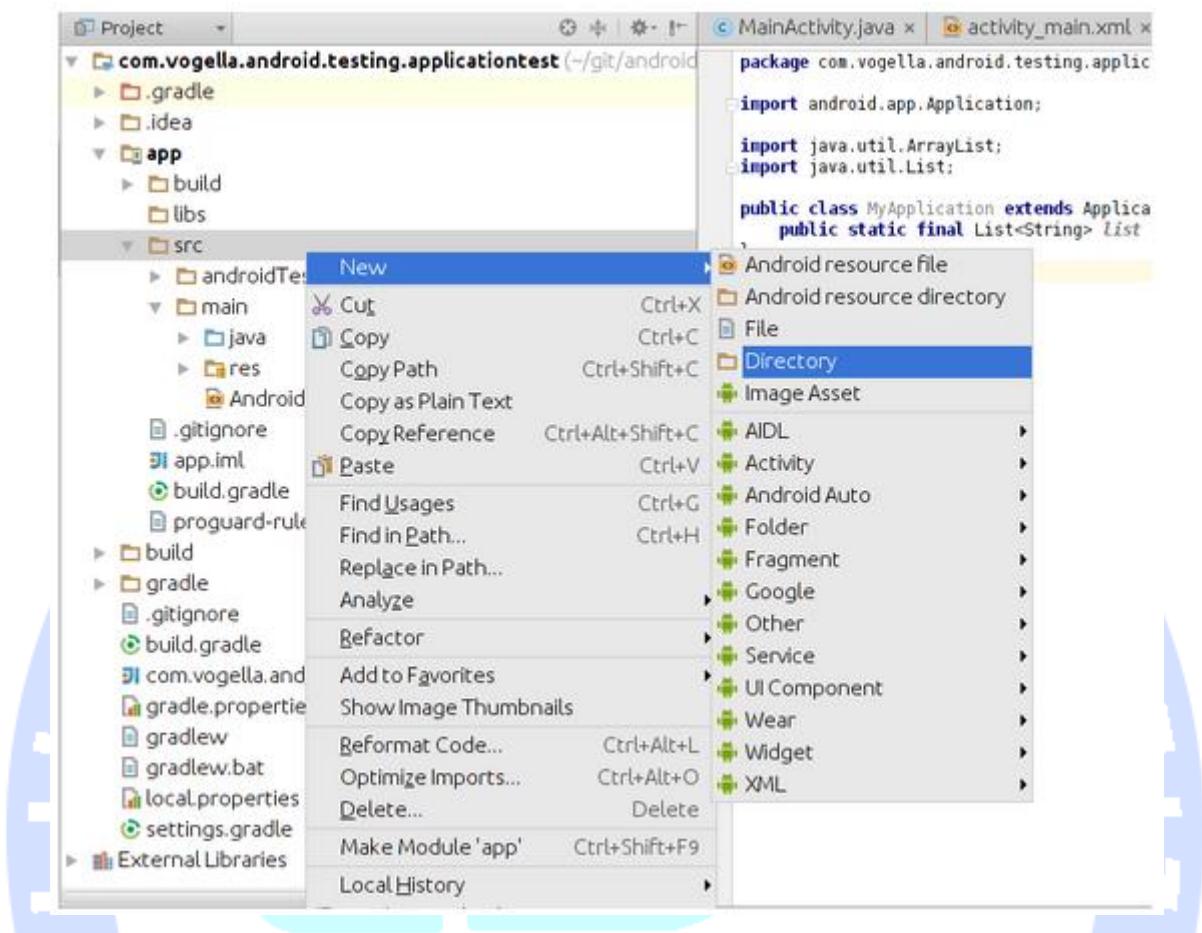


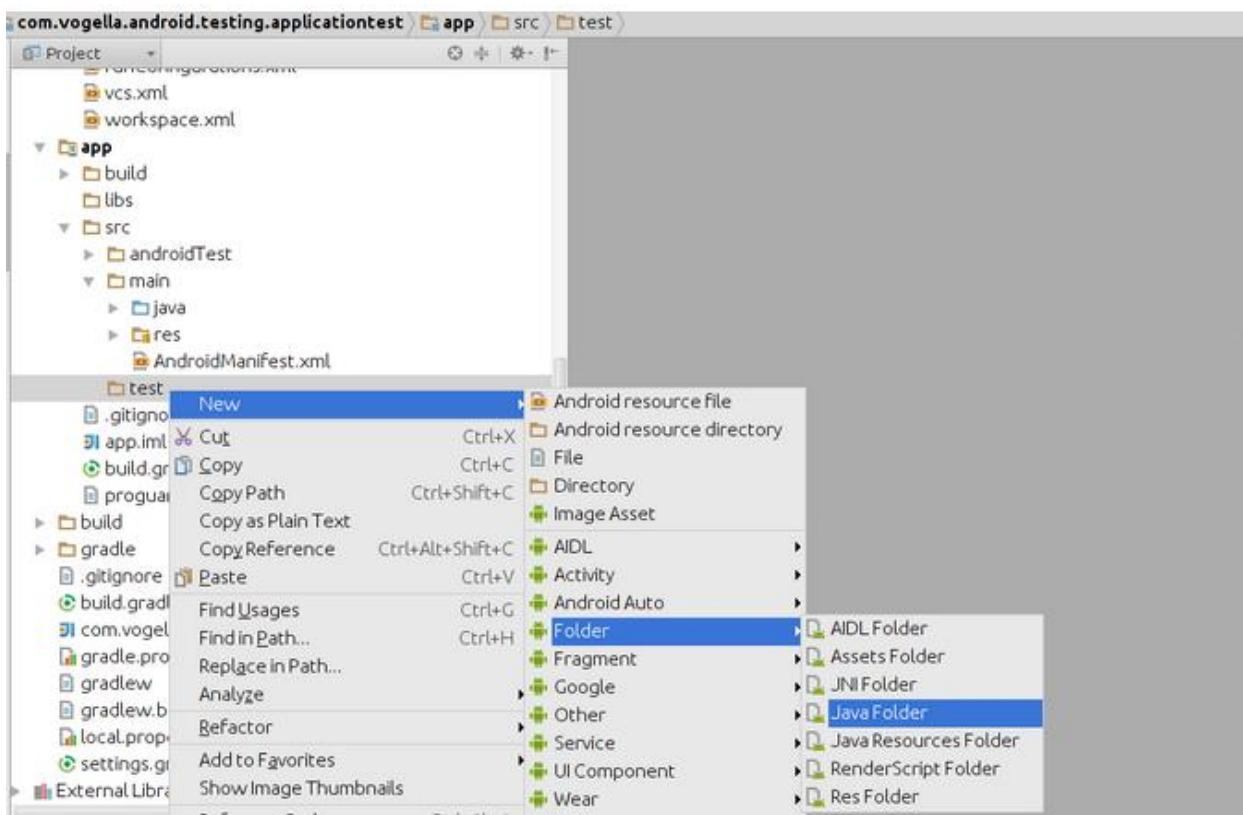
## Android Studio 19-7 ساخت پوشه‌ی تست در محیط کاری

ویرایش های جدید محیط برنامه نویسی Android Studio، یک پوشه ی تست (test folder) به قالب آماده ی پروژه ی پیش فرض (project template) خود اضافه کرده است. در صورت استفاده از قالبی که قابلیت ایجاد پوشه ی تست را به صورت درون ساخته ندارد، بایستی این پوشه را خود به صورت دستی ایجاد نمایید. جهت ایجاد پوشه ی تست در محیط کاری مذکور، قادر Project را باز نمایید. در این view می توانید ساختار درختی و directory structure پروژه ی خود را مشاهده نمایید.



Src را انتخاب نموده و پس از باز کردن Context menu یک پوشه ی تست (test folder) جدید ایجاد نمایید.



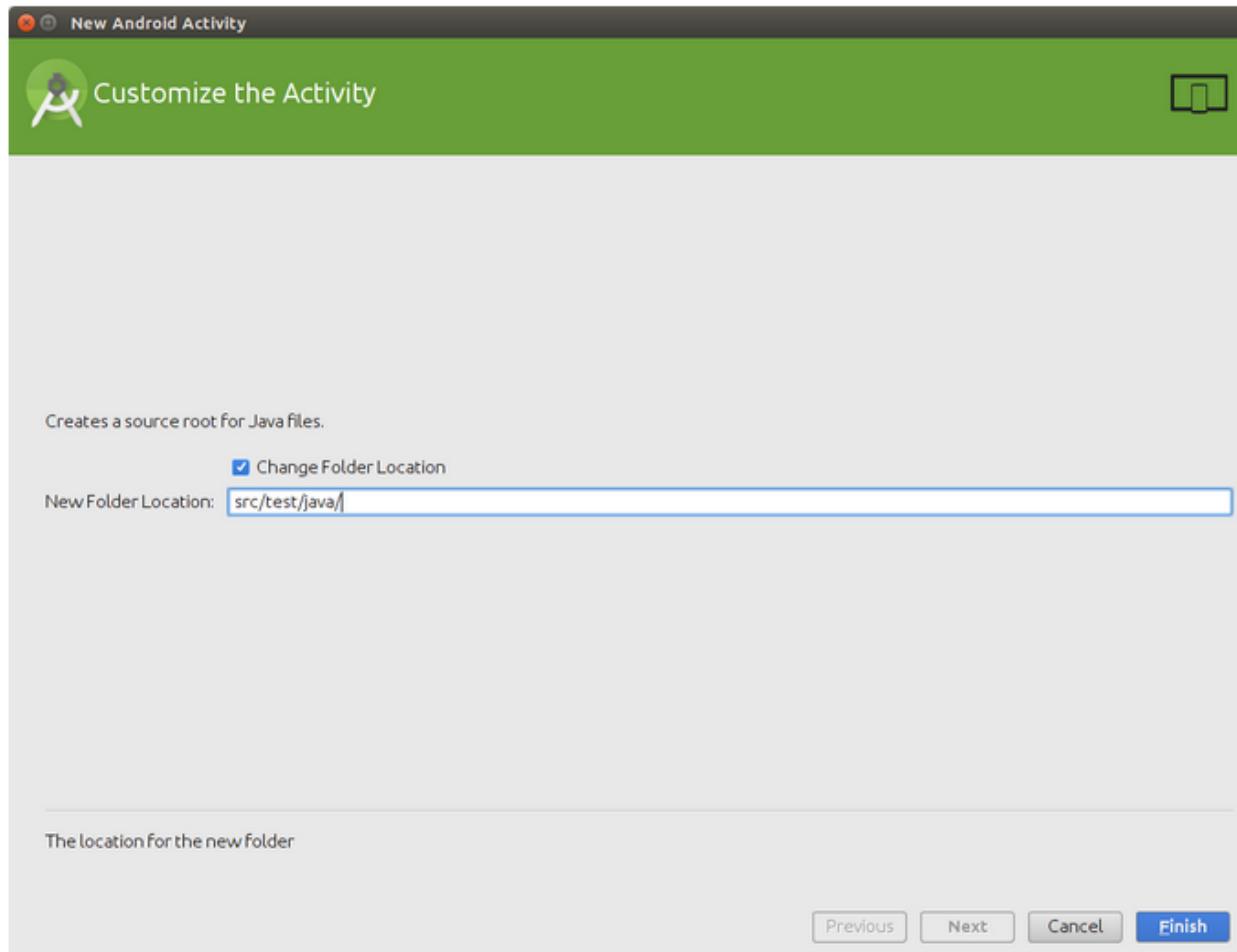


## آموزشگاه تحلیلکر داده

694

آدرس آموزشگاه : تهران - خیابان شریعتی - بالاتر از خیابان ملک - جنب بانک صادرات - پلاک 651 طبقه دوم - واحد 7

88146323 - 88446780 - 88146330



در صورتی که همچنان کامل انجام نشده باشد، کتابخانه ی JUnit (dependency) را به فایل `gradle.build` اضافه نمایید.

```
dependencies {
    // Unit testing dependencies
    testCompile 'junit:junit:4.12'
    // Set this dependency if you want to use the Hamcrest matcher library
    testCompile 'org.hamcrest:hamcrest-library:1.3'
    // more stuff, e.g., Mockito
}
```

**توجه:** لازم به ذکر است که در نتیجه ی ساخت پوشه ی Java، ممکن است پوشه ی جدید تست به عنوان `source file` به فایل `gradle.build` اضافه گردد. چنانچه دستور زیر در فایل

app/build.gradle موجود باشد، در آن صورت می بایست کد ذکر شده را کاملا از فایل حذف نمایید. تست نباید به هیچ وجه به عنوان یک source folder معمولی در نظر گرفته شود.

```
sourceSets { main { java.srcDirs = ['src/main/java', 'src/test/java/] } }
```

پس از آن کافی است unit test خود را به این ساختار درختی پوشش دهید. اضافه نمایید.



## بخش سوم :

# اجرای unit test بر روی اپلیکیشن با استفاده از فریم ورک Mockito تست گیری

آموزش حاضر نحوه‌ی تست نویسی برای نرم افزارهای اندرویدی را با استفاده از فریم ورک Mockito شرح می‌دهد.

## 20-7- تست بخش‌های مجازی نرم افزار با استفاده از آبجکت‌های ساختگی (mock objects)

20-7- هدف از نوشتن unit test و چالش‌هایی که در تست با آن‌ها مواجه می‌شوید

Unit test، همان طور که از اسم آن پیدا است، بایستی یک اپلیکیشن را در سطح کلاس‌های مجزا آزمایش نماید، به بیان دیگر هر یک از کلاس‌های تشکیل دهنده‌ی پروژه را به صورت جداگانه مورد تست قرار داده و از عملکرد صحیح تمامی آن‌ها اطمینان حاصل کند. برای نیل به این هدف در unit test، لازم است side effect (عوارض) یا وابستگی به سایر کلاس‌ها و بخش‌های سیستم حذف شود. بدیهی است که برای حذف این وابستگی‌ها، می‌بایست dependency به دیگر کلاس‌ها را حذف نمایید.

این کار با استفاده از جایگزین برای dependency‌های واقعی پروژه امکان پذیر می‌باشد.

## 7-20-2-طبقه بندی کلاس های تست گیری مختلف

یک آبجکت خام بین بخش های مختلف پروژه پاس داده می شود اما هیچگاه توابع داخل آن به معنای واقعی فراخوانی نمی شوند (dummy object): آبجکتی که API و توابع کتابخانه ای یک آبجکت را عیناً کپی نماید. در واقع این آبجکت ها به توسعه دهنده اجازه می دهند تا آبجکت هایی که کلاس های پروژه به آن نیاز دارند را، بدون اینکه پیاده سازی و رفتار خاصی برای آن تعریف کرده باشند، شبیه سازی نماید. با استفاده از این آبجکت می توان، به عنوان مثال، لیست پارامترهای یک متده را با مقادیر پر کرد.

Fake object ها / توابع ساختگی دارای پیاده سازی و قابلیت های واقعی اما معمولاً بسیار ساده هستند. به عنوان مثال، بجای استفاده از دیتابیس واقعی از یک دیتابیس مقیم در حافظه (in-memory) استفاده می کنند.

پیش از توضیح مفهوم stub class، بد نیست به شرح مفهوم test stub بپردازیم. یک Test stub یک قطعه برنامه است که قابلیت های ضروری (که کامپوننت مورد تست به آن ها وابسته هست) از کامپوننت مورد تست را شبیه سازی می کند. test stub ها در جواب توابع صدا زده شده در طول تست، پاسخ های معین ارائه می دهند.

به عبارت دیگر، stub ها برنامه هایی هستند که به منزله یک جایگزین موقتی برای کامپوننت فراخوانی شده ایفا نقش کرده و همان خروجی نرم افزار یا برنامه ی واقعی را تولید می کنند.

Stub class (کلاس جایگزین با پیاده سازی نیمه)، عبارت است از implementation ناتمام از کل یک کلاس یا interface که نمونه ای از آن در طول تست گیری مورد استفاده قرار می گیرد. stub ها ممکن است اطلاعاتی درباره ی فراخوانی های رخداده نیز ثبت نمایند.

حال به شرح مفهوم mock object (آبجکت ساختگی) می پردازیم. Mock object ابجکت های ساختگی هستند که رفتار آبجکت های واقعی را به صورت کنترل شده و مشخص شبیه سازی و به عبارتی تقلید می کنند. برنامه نویس اغلب با استفاده از آبجکت ساختگی، سعی دارد تا رفتار آبجکت (واقعی) دیگری را شبیه سازی نماید.

به تعریف دیگر، عبارت است از پیاده سازی ساختگی برای یک interface یا کلاس به در آن خروجی فراخوانی برخی توابع تعریف می شود.

می توان test double ها را به آبجکت های مورد تست ارسال کرد (یک واژه عمومی برای تمامی توابع و آبجکت های ساختگی است که به منظور تست نرم افزار و کنترل کیفیت آن مورد استفاده قرار می گیرند). تست های شما می توانند بررسی کنند آیا کلاس مورد نظر در طول اجرای تست طبق انتظار پاسخ می دهند یا خیر.

به طور مثال شما در طول تست می توانید بررسی کنید، آیا توابع مورد نظر در سطح آبجکت به درستی فراخوانی می شوند یا خیر. این قابلیت به شما امکان می دهد تا اطمینان حاصل نمایید به هنگام تست گیری، منحصراً کلاس تست شده و اینکه تست های شما وابستگی به بخش های دیگر برنامه نداشته باشند (یا در نتیجه اجرای تست سایر بخش های پروژه تاثیر نپذیرند و عوارض جانبی درکار نباشد).

توسعه دهنده این امکان را دارد که آبجکت های ساختگی (mock) را به صورت اختصاصی تنظیم کند. گفتنی است که آبجکت های ساختگی به کد و تنظیمات کمتری احتیاج داشته و بایستی در اجرای تست بر روی نرم افزار بر سایر روش ها ارجحیت داشته باشند.

### 3-20-7- نحوه ساخت mock object

می توانید آبجکت های ساختگی را خود به صورت دستی (از طریق کد) ایجاد کنید یا اینکه از یک فریم ورک شبیه سازی (mocking framework) جهت تقلید قابلیت ها و رفتارهای کلاس های واقعی اپلیکیشن استفاده نمایید. Mocking framework ها به شما این امکان را می دهند تا آبجکت های ساختگی ایجاد نموده و برای آن ها رفتاری مشابه رفتار کلاس مورد تست تعریف نمایید.

مثالی شناخته شده و معمول از data provider، یک mock object ساده هست. در مرحله ی استفاده از اپلیکیشن، برای اتصال به data source، پیاده سازی و کد

لازم مورد استفاده قرار می گیرد. این در حالی است که برای تست، یک آبجکت ساختگی منبع داده ای را شبیه سازی نموده و اطمینان حاصل می نماید که شرایط تست همواره ثابت هستند.

این آبجکت های ساختگی را می توان در اختیار کلاس مورد تست قرار داد. به همین جهت، کلاسی که تست می شود باید وابستگی (dependency) زیادی به داده های خارجی داشته باشد.

فریم ورک های شبیه سازی (Mocking framework) در کل این امکان را فراهم می آورند تا تعامل مورد انتظار با آبجکت ساختگی (mock) را به راحتی تست نمایید. به عنوان مثال، شما می توانید بررسی کنید که تنها متدهای مدنظر بر روی آبجکت ساختگی فراخوانی شده اند.

#### 7-20-4-استفاده از Mockito برای شبیه سازی و ایجاد آبجکت ساختگی

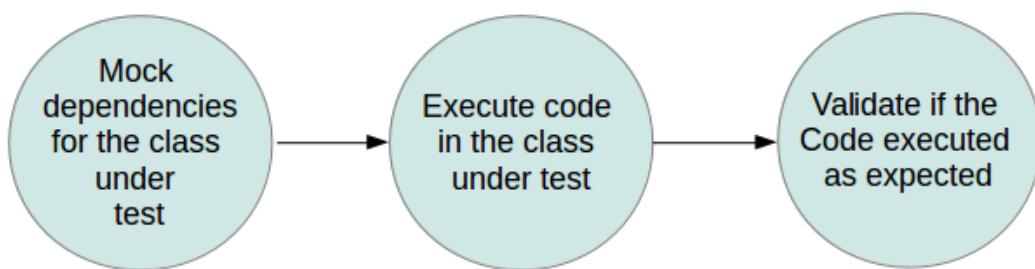
یک فریم ورک پرطرفدار شبیه سازی و ایجاد آبجکت ساختگی برای تست نرم افزار است که گاه همراه با JUnit نیز مورد استفاده قرار می گیرد. این فریم ورک توسعه دهنده را قادر می سازد آبجکت های ساختگی ایجاد کرده و رفتار یا پیاده سازی آن ها را بر اساس نیاز تنظیم کند. در واقع توسعه دهنده با بهره گیری از Mockito می تواند تست نویسی برای کلاس هایی که به سایر کلاس ها و کتابخانه ها وابسته هستند یا به عبارتی دیگر dependency به داده های خارجی دارند نماید.

آسان

بسیار

در استفاده از Mockito در تست های نرم افزاری خود، اغلب اقداماتی نظیر عنوان شده در زیر را انجام خواهید داد:

- تمامی Dependency ها به خارج از کلاس مورد تست (کلاس هایی که کلاس مورد تست با آن ها تعامل دارد) را شبیه سازی نموده و سپس آبجکت های ساختگی خود را داخل کد مورد تست درج می نمایید.
- کد مورد تست را اجرا نمایید.
- اطمینان حاصل نمایید که کد مورد نظر به درستی اجرا می شود.



## ۲۱-۷-افزودن dependency در قالب Mockito به پروژه

### ۲۱-۷-۱-استفاده از سیستم کامپایل Gradle

برای کامپایل پروژه با استفاده از Gradle، لازم است ابتدا کتابخانه (dependency) زیر را به فایل Gradle build اضافه نمایید.

```
repositories { jcenter() }
dependencies { testCompile "org.mockito:mockito-core:2.0.57-beta" }
```

### ۲۱-۷-۲-استفاده از سیستم کامپایل Maven

کاربران سیستم Maven می توانند خود یک dependency تعريف کنند. عبارت های `a:"mockito-core"`, `g:"org.mockito"` جستجو <http://search.maven.org> را در وب سایت جستجو نموده و فایل pom مورد نیاز را پیدا نمایید.

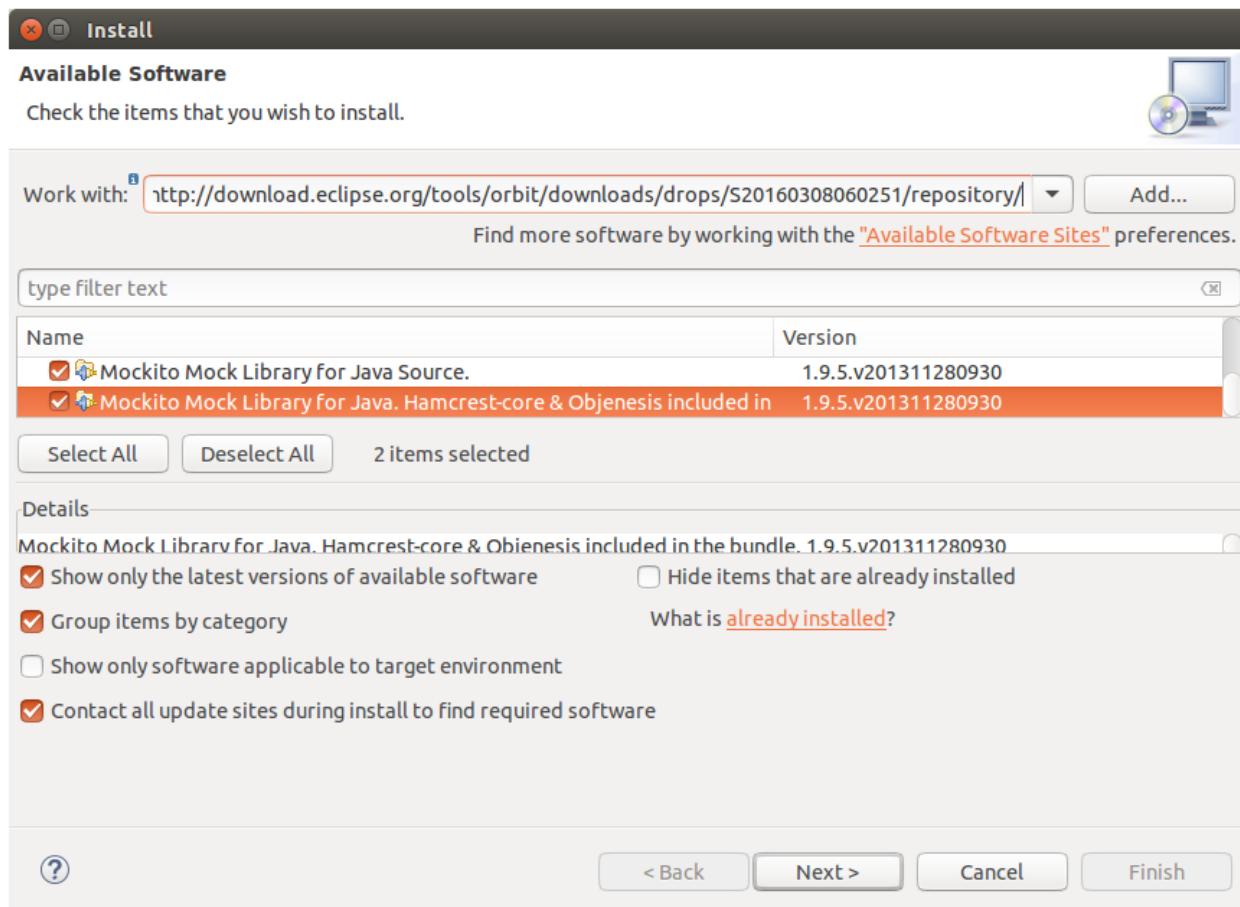
### ۲۱-۷-۳-استفاده از محیط کاری Eclipse

محیط برنامه نویسی Eclipse با هر دو سیستم Gradle و Maven سازگار می باشد. لازم به ذکر است که آخرین ویرایش فریم ورک Mockito امکان دانلود ابزار هر دو سیستم را نمی دهد. بنابراین توصیه می شود در این محیط کاری از مجموعه ابزار یکی از دو سیستم مزبور استفاده نمایید.

## 7-21-4-افزونه توییسی برای OSGi با Eclipse (چارچوب توسعه و نصب کتابخانه ها و کامپوننت های نرم افزاری)

در اپلیکیشن های RCP یا برنامه های تحت وب سمت کلاینت مبتنی بر Eclipse، کتابخانه ها و Repository های مورد نیاز غالبا از مخزن پروژه‌ی p2 repository گرفته می‌شوند. dependency های Orbit هم منبع خوبی برای کتابخانه های کمکی (third party) هستند که به راحتی در اپلیکیشن ها یا افزونه های مبتنی بر Eclipse می‌توانید از آن ها استفاده نمایید.

برای دسترسی به آدرس دسترسی به آدرس <http://download.eclipse.org/tools/orbit/downloads> مراجعه نمایید.



## 22-7-استفاده از توابع کتابخانه‌ای Mockito API / Mockito

### 22-7-1-دستورات Static import

اگر به دستور `static import org.mockito.*;` اضافه نمایید، در آن صورت قادر خواهید بود به متدهای `Mockito.mock()` به طور مستقیم دسترسی داشته باشید. `static import` ها به شما این امکان را می‌دهند تا اعضای `static` کلاس همچون متدها و فیلدها را بدون اینکه اسم کلاس را به طور صریح ذکر کنید، فراخوانی و استفاده نمایید.

## 7-22-2- اعلان و تنظیم آبجکت های ساختگی / mock objects با فریم ورک Mockito

همان طور که قبلاً نیز ذکر شد، Mockito یک فریم ورک است که امکان تعریف آبجکت های ساختگی (mock object) را فراهم می‌آورد. برای این منظور، فریم ورک نام بردۀ دو روش زیر را ارائه می‌دهد:

1. فراخوانی تابع `mock()`

2. استفاده از دستور `@Mock`

در صورت استفاده از روش دوم، شما با استناد خود آبجکت های ساختگی را مقداردهی اولیه نمایید. جهت مقداردهی اولیه این آبجکت لازم است از `MockitoRule` استفاده کنید. `MockitoRule` متد `static` ای به نام `MockitoAnnotations.initMocks(this)` را فراخوانی می‌کند که فیلد های نشانه گذاری شده (دارای `annotation`) را مقداردهی می‌نماید. در صورت تمایل می‌توانید از دستور `@RunWith(MockitoJUnitRunner.class)` استفاده کرده و همین نتیجه را بدست بیاورید.

کد زیر استفاده از دستور `@Mock` و `MockitoRule` را به صورت عملی نمایش می‌دهد.

```
import static org.mockito.Mockito.*;
public class MockitoTest {
    @Mock
    MyDatabase databaseMock;
    @Rule public MockitoRule mockitoRule = MockitoJUnit.rule();
    @Test
    public void testQuery() {
        ClassToTest t = new ClassToTest(databaseMock);
        boolean check = t.query("* from t");
        assertTrue(check);
        verify(databaseMock).query("* from t");
    }
}
```

1. به Mockito اعلان می‌کند که از نمونه `databaseMock` شبیه سازی یا `mock` کند.
2. به Mockito دستور می‌دهد تا بر مبنای دستور `@Mock`، آبجکت های ساختگی یا `mock` ایجاد کند.

3. با استفاده از آبجکت ساختگی از کلاس مورد تست نمونه سازی (instantiate) می کند.
4. بخشی از کد کلاس مورد تست را اجرا می کند.
5. انتظار دارد (assert می کند) که متده فراخوانده شده در خروجی مقدار بولی true را باز گرداند.
6. بررسی کرده و اطمینان حاصل می کند که متده query بر روی mock یا نمونه ی ساختگی از MyDatabase فراخوانی می شود.

### 7-22-3- تنظیم آبجکت های ساختگی / mock ها

می توانید به طور دقیق مشخص کنید که یک متده پس از فراخوانی، چه مقداری را در خروجی بازگرداند و این کار را به واسطه ی تکنیک نوشتن روان، خوانا و پشت سرهم دستورات یا همان که فریم ورک Mockito در اختیار شما قرار می دهد، انجام دهید.

دو متده (....).thenReturn(....) که به صورت زنجیره ای و پشت سرهم فراخوانی می شوند این امکان را به شما می دهد تا یک شرط معین تعریف کرده و مقدار بازگشته آن را نیز مشخص نمایید. اگر بیش از یک مقدار مشخص کنید، این مقادیر دقیقا به همان ترتیبی که تعریف شده اند، برگردانده می شوند. Mockito ها ممکن است بر اساس پارامترهای ارسالی به متده، مقادیر متفاوتی را بازگردانی نمایند. همچنین می توانید با استفاده از توابع anyInt anyString یا anyLong تعیین کنید که مستقل از مقدار ورودی، مقدار خاصی به عنوان خروجی از متده بازگردانده شود.

```
import static org.mockito.Mockito.*;
import static org.junit.Assert.*;
@Test
public void test1() {
    // create mock
    MyClass test = Mockito.mock(MyClass.class);
    // define return value for method getUniqueld()
    when(test.getUniqueld()).thenReturn(43);
    // use mock in test....
    assertEquals(test.getUniqueld(), 43);
}
// Demonstrates the return of multiple values
@Test
public void testMoreThanOneReturnValue() {
    Iterator i= mock(Iterator.class);
    when(i.next()).thenReturn("Mockito").thenReturn("rocks");
    String result=i.next()+" "+i.next();
```

```

//assert
assertEquals("Mockito rocks", result);
}
// this test demonstrates how to return values based on the input
@Test
public void testReturnValueDependentOnMethodParameter() {
    Comparable c= mock(Comparable.class);
    when(c.compareTo("Mockito")).thenReturn(1);
    when(c.compareTo("Eclipse")).thenReturn(2);
    //assert
    assertEquals(1,c.compareTo("Mockito"));
}
// this test demonstrates how to return values independent of the input value
@Test
public void testReturnValueIndependentOnMethodParameter() {
    Comparable c= mock(Comparable.class);
    when(c.compareTo(anyInt())).thenReturn(-1);
    //assert
    assertEquals(-1 ,c.compareTo(9));
}
// return a value based on the type of the provide parameter
@Test
public void testReturnValueIndependentOnMethodParameter() {
    Comparable c= mock(Comparable.class);
    when(c.compareTo(isA(Todo.class))).thenReturn(0);
    //assert
    Todo todo = new Todo(5);
    assertEquals(todo ,c.compareTo(new Todo(1)));
}

```

دو تابع `doReturn(...).when(...).methodCall` نیز که به صورت پشت سرهم (زنگیره ای/chained) صدا زده می شوند مشابه نمونه‌ی قبلی عمل می کنند، اما بیشتر برای متدهایی که خروجی ندارند `void` هستند، کارا هستند. می توان با استفاده از `doThrow` برای متدهایی که خروجی ندارند `void` بر می گردانند، یک خطا (exception) صادر کرد. این کاربرد در تکه کد زیر به نمایش گذاشته می شود.

```

import static org.mockito.Mockito.*;
import static org.junit.Assert.*;
// this test demonstrates how use doThrow
@Test(expected=IOException.class)
public void testForIOException() {
    // create an configure mock
    OutputStream mockStream = mock(OutputStream.class);
    doThrow(new IOException()).when(mockStream).close();
    // use mock
    OutputStreamWriter streamWriter= new OutputStreamWriter(mockStream);
    streamWriter.close();
}

```

}

## 7-22-4-متد verify() (بررسی صحت فرآخوانی یک متدها با پارامترهای مورد نظر)

Mockito تمامی توابع فرآخوانده شده و پارامترهای ارسالی به آبجکت ساختگی (mock) را رصد کرده و اطلاعات مربوط به آن ها را ثبت می کند. شما می توانید با استفاده از متد verify() بر روی آبجکت ساختگی مطمئن شوید که شرایط و نیاز های تعیین شده برآورده شده اند. به طور مثال، می توانید بررسی کرده و اطمینان حاصل نمایید که یک متدها با پارامترهای مورد نیاز فرآخوانی شده اند یا خیر. این تست گیری در اصطلاح behavior testing خوانده می شود. متد معینی testing با نتیجه ای متد فرآخوانی شده کاری ندارد، بلکه می خواهد مطمئن شود آیا متد معینی با پارامترهای مورد نظر فرآخوانی می شود یا خیر.

```
import static org.mockito.Mockito.*;
@Test
public void testVerify() {
    // create and configure mock
    MyClass test = Mockito.mock(MyClass.class);
    when(test.getUniqueId()).thenReturn(43);
    // call method testing on the mock with parameter 12
    test.testing(12);
    test.getUniqueId();
    test.getUniqueId();
    // now check if method testing was called with the parameter 12
    verify(test).testing(Mockers.eq(12));
    // was the method called twice?
    verify(test, times(2)).getUniqueId();
    // other alternatives for verifying the number of method calls for a method
    verify(mock, never()).someMethod("never called");
    verify(mock, atLeastOnce()).someMethod("called at least once");
    verify(mock, atLeast(2)).someMethod("called at least twice");
    verify(mock, times(5)).someMethod("called five times");
    verify(mock, atMost(3)).someMethod("called at most 3 times");
}
```

## 7-22-5-قرار دادن آبجکت های جاوا در Spy (رصد و شنود آبجکت های جاوا و توابع فراخوانده شده بر روی آن با spy)

می توانید با درج دستور `@Spy` یا فراخوانی متدهای آبجکت (spy) آبجکت های (واقعی) جاوا را رصد نمایید. بدین معنی که ابتدا آبجکت را داخل دو دستور ذکر شده قرار دهید، سپس تمامی توابع متعارف آبجکت را صدا زده و در این میان هر اتفاق یا تعاملی که رخداد را دقیقاً ردگیری (شنود و دنبال) کنید. در واقع با این کار تمامی توابع، به جز آن هایی که خود به صورت صریح مستثنی نمایید، بر روی این آبجکت صدا زده می شوند.

```
import static org.mockito.Mockito.*;
// Lets mock a LinkedList
List list = new LinkedList();
List spy = spy(list);
//You have to use doReturn() for stubbing
doReturn("foo").when(spy).get(0);
// this would not work
// real method is called so spy.get(0)
// throws IndexOutOfBoundsException (list is still empty)
when(spy.get(0)).thenReturn("foo");
```

متدهای `verifyNoMoreInteractions()` به شما این امکان را می دهد تا مطمئن شوید هیچ تابع دیگری (به غیر از آنچه خود به صورت صریح مشخص کرده اید) فراخوانی نشده است.

## 7-22-6-پیاده سازی الگوی dependency injection در فریم ورک @InjectMocks با استفاده از دستور Mockito

فریم ورک Mockito با ارائه دستوری به نام `@InjectMocks` به شما این امکان را می دهد تا بر اساس نوع dependency مورد نظر، آن را از طریق تابع سازنده (constructor)، متدهای فیلد یا فیلد تزریق نمایید (الگوی dependency injection را به این صورت پیاده سازی کنید).

```
public class ArticleManager {
    private User user;
    private ArticleDatabase database;
    ArticleManager(User user) {
```

```

        this.user = user;
    }
    void setDatabase(ArticleDatabase database) { }
}

```

این کلاس را می توان از طریق Mockito ساخته و کلاس های مورد نیاز یا dependency های آن را با استفاده از آبجکت های ساختگی (mocks) به داخل آن تزریق کرد.

```

@RunWith(MockitoJUnitRunner.class)
public class ArticleManagerTest {
    @Mock ArticleCalculator calculator;
    @Mock ArticleDatabase database;
    @Most User user;
    @Spy private UserProvider userProvider = new ConsumerUserProvider();
    @InjectMocks private ArticleManager manager; (1)
    @Test public void shouldDoSomething() {
        // assume that ArticleManager has a method called initialize which calls a method
        // addListener with an instance of ArticleListener
        manager.initialize();
        // validate that addListener was called
        verify(database).addListener(any(ArticleListener.class));
    }
}

```

یک نمونه از کلاس ArticleManager ساخته و آبجکت های ساختگی (mock ها) را به داخل آن تزریق می کند.

## 7-22-7- دسترسی به آرگومان های ارسالی به توابع (arguments)

کلاس ArgumentCaptor به شما امکان می دهد تا به آرگومان های پاس داده به متدها در طول پروسه ای بررسی صحت فراخوانی متدها را داشته باشید. بدین وسیله شما قادر خواهید بود به آرگومان های توابع فراخوانی شده دسترسی داشته و سپس آن ها را در تست های خود مورد استفاده قرار دهید.

```

import static org.hamcrest.Matchers.hasItem;
import static org.junit.Assert.assertThat;
import static org.mockito.Mockito.mock;
import static org.mockito.Mockito.verify;
import java.util.Arrays;
import java.util.List;
import org.junit.Rule;
import org.junit.Test;
import org.mockito.ArgumentCaptor;
import org.mockito.Captor;

```

```

import org.mockito.junit.MockitoJUnit;
import org.mockito.junit.MockitoRule;
public class MockitoTests {
    @Rule public MockitoRule rule = MockitoJUnit.rule();
    @Captor
    private ArgumentCaptor<List<String>> captor;
    @Test
    public final void shouldContainCertainListItem() {
        List<String> asList = Arrays.asList("someElement_test", "someElement");
        final List<String> mockedList = mock(List.class);
        mockedList.addAll(asList);
        verify(mockedList).addAll(captor.capture());
        final List<String> capturedArgument = captor.<List<String>>getValue();
        assertThat(capturedArgument, hasItem("someElement"));
    }
}

```

## 7-22-8- محدودیت های فریم ورک Mockito

فریم ورک Mockito نیز مانند سایر چارچوب های کاری محدودیت هایی دارد. این فریم ورک قادر به تست ساختارهای زیر نمی باشد:

- کلاس هایی که به صورت final تعریف شده باشند و امکان ارث بری از آن ها وجود نداشته باشد
- کلاس های anonymous یا کلاس های فاقد اسم
- انواع داده ای اولیه و primitive همچون string

## 7-23- استفاده از فریم ورک Mockito در اندروید

Mockito را می توان به طور مستقیم نیز در unit test های اندروید مورد استفاده قرار داد. کافی است dependency مربوطه را به فایل Gradle build اضافه نمایید. جهت استفاده از این فریم ورک در تست های اندروید، لازم است instrumentation و dexmaker-mockito را در dependencies فایل Gradle build اضافه نمایید.

```
dependencies {
```

```

    testCompile 'junit:junit:4.12'
    // required if you want to use Mockito for unit tests
    testCompile 'org.mockito:mockito-core:1.+'
    // required if you want to use Mockito for Android instrumentation tests
    androidTestCompile 'org.mockito:mockito-core:1.+'
    androidTestCompile "com.google.dexmaker:dexmaker:1.2"
    androidTestCompile "com.google.dexmaker:dexmaker-mockito:1.2"
}

```

## تمرین: نوشتن تست مبتنی بر instrumentation با استفاده از فریم

### ورک Mockito

#### ایجاد اپلیکیشن های آزمایشی (مورد تست) در اندروید

یک اپلیکیشن اندروید جدید با نام `apk` ایجاد کنید. حال یک متدهای `static Intent createQuery(Context context, String query, String value)` در `com.vogella.android.testing.mockito.contextmock` تعریف نمایید. حال یک آبجکت `Intent` با پارامترهای زیر ایجاد نمایید.

```

public static Intent createQuery(Context context, String query, String value) {
    // Reuse MainActivity for simplification
    Intent i = new Intent(context, MainActivity.class);
    i.putExtra("QUERY", query);
    i.putExtra("VALUE", value);
    return i;
}

```

#### 7-23- اضافه کردن مورد نیاز Mockito به فایل app/build.gradle

```

dependencies {
    // the following is required to use Mockito and JUnit for your
    // instrumentation unit tests on the JVM
    androidTestCompile 'junit:junit:4.12'
    androidTestCompile 'org.mockito:mockito-core:2.0.57-beta'
    androidTestCompile 'com.android.support.test:runner:0.3'
}

```

```

    androidTestCompile "com.google.dexmaker:dexmaker:1.2"
    androidTestCompile "com.google.dexmaker:dexmaker-mockito:1.2"
    // the following is required to use Mockito and JUnit for your unit
    // tests on the JVM
    testCompile 'junit:junit:4.12'
    testCompile 'org.mockito:mockito-core:1.+'
}

```

## 7-23-2-ساخت یک unit test جدید

اکنون یک unit test جدید با استفاده از Mockito ایجاد نموده و بررسی کنید آیا intent با داده های extra مرتبط فراخوانی می شود یا خیر.

برای این منظور آبجکت Context را با استفاده از توابع Mockito به صورت زیر شبیه سازی نمایید.

```

package com.vogella.android.testing.mockitocontextmock;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.mockito.Mockito;
import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertNotNull;
public class TextIntentCreation {
    @Test
    public void testIntentShouldBeCreated() {
        Context context = Mockito.mock(Context.class);
        Intent intent = MainActivity.createQuery(context, "query", "value");
        assertNotNull(intent);
        Bundle extras = intent.getExtras();
        assertNotNull(extras);
        assertEquals("query", extras.getString("QUERY"));
        assertEquals("value", extras.getString("VALUE"));
    }
}

```

**تمرین: ساخت آبجکت های ساختگی (mock) با استفاده از توابع**

**Mockito**

**هدف از تمرین**

در تمرین حاضر، ابتدا API ای تعییه نمایید که قابل تست و شبیه سازی باشند. سپس با استفاده از توابع Mockito آن ها را تست نمایید.

## 7-23-3- ایجاد یک نمونه API Twitter

ابتدا یک کلاس TwitterClient پیاده سازی کنید که از نمونه ای از اینترفیس ITweet استفاده کرده و توابع آن را فراخوانی می کند. دسترسی به نمونه های این interface ملال آور بوده و نیازمند استفاده از یک سرویس هزینه بر و پیچیده می باشد. علاوه بر این، سرویس مذکور بایستی برای دسترسی، اول فراخوانی و اجرا شود.

```
public interface ITweet {
    String getMessage();
}

public class TwitterClient {
    public void sendTweet(ITweet tweet) {
        String message = tweet.getMessage();
        // send the message to Twitter
    }
}
```

## 7-23-4- شبیه سازی از نمونه API

برای اینکه لازم نباشد جهت دسترسی به نمونه هایی از اینترفیس ITweet، یک سرویس هزینه بر و پیچیده راه اندازی نمایید، توصیه می شود نمونه ای interface مزبور را به واسطه ای فریم ورک Mockito شبیه سازی کنید.

```
@Test
public void testSendingTweet() {
    TwitterClient twitterClient = new TwitterClient();
    ITweet iTweet = mock(ITweet.class);
    when(iTweet.getMessage()).thenReturn("Using Mockito is great");
    twitterClient.sendTweet(iTweet);
}
```

اکنون کلاس TwitterClient می تواند توابع اینترفیس ساختگی و شبیه سازی شده ای ITweet را پیاده سازی نماید. در نتیجه ای استفاده از توابع این interface شبیه سازی شده (فراخوانی متدهای ساختگی) بر روی نمونه ای ساختگی ITweet، پیغام "Using Mockito is great" به عنوان getMessage() خروجی متدهای اختیار شما قرار می گیرد.

## 7-23-بررسی صحت فراخوانی متدهای

اطمینان حاصل نمایید که متدهای `getMessage()` حداقل یکبار فراخوانی می شود.

```
@Test
public void testSendingTweet() {
    TwitterClient twitterClient = new TwitterClient();
    ITweet iTweet = mock(ITweet.class);
    when(iTweet.getMessage()).thenReturn("Using Mockito is great");
    twitterClient.sendTweet(iTweet);
    verify(iTweet, atLeastOnce()).getMessage();
}
```

## 7-23-مرحله‌ی اعتبار سنجی تست

تست را اجرا کرده و مطمئن شوید که با موفقیت انجام می شود.

## 7-24-استفاده از Mockito به همراه PowerMock

### 7-24-1-استفاده از PowerMock برای شبیه سازی متدهای static

فریم ورک Mockito قادر به شبیه سازی متدهای static نیست. برای شبیه سازی رفتار متدهای static ، بایستی از PowerMock استفاده نمایید. PowerMock یک کلاس به نام "PowerMockito" ارائه می دهد که برای ساخت `class/object/mock`، بررسی صحت اجرا(اعتبارسنجی) و برآورده ساختن انتظارات این کلاس را فراخوانی می کند.

برای بررسی صحت برآورده شدن انتظارات می توانید همچنان از خود Mockito (توابع `times()`، `anyInt()` استفاده نمایید.

```
import java.net.InetAddress;
import java.net.UnknownHostException;
public final class NetworkReader {
```

```

public static String getLocalHostname() {
    String hostname = "";
    try {
        InetAddress addr = InetAddress.getLocalHost();
        // Get hostname
        hostname = addr.getHostName();
    } catch ( UnknownHostException e ) {
    }
    return hostname;
}

```

برای نوشتن تستی که قابلیت های NetworkReader را برای شما شبیه سازی می کند، می توانید از تکه کد زیر استفاده نمایید.

```

import org.junit.runner.RunWith;
import org.powermock.core.classloader.annotations.PrepareForTest;
@RunWith( PowerMockRunner.class )
@PrepareForTest( NetworkReader.class )
public class MyTest {
// Find the tests here
@Test
public void testSomething() {
    mockStatic( NetworkUtil.class );
    when( NetworkReader.getLocalHostname() ).thenReturn( "localhost" );
    // now test the class which uses NetworkReader
}

```

## 25-7-استفاده از یک wrapper بجای

همچنین شما می توانید تابع static دلخواه خود را داخل یک wrapper گنجانده (یک تابع که تابع دیگری را فراخوانی می کند) و سپس آن را توسط Mockito شبیه سازی نمایید.

```

class FooWrapper {
    void someMethod() {
        Foo.someStaticMethod()
    }
}

```



بخش چهارم :

## استفاده از فریم ورک Hamcrest جهت اجرای تست نرم افزاری

این مبحث به اجرای تست نرم افزاری بر روی اپلیکیشن به وسیلهٔ فریم ورک Hamcrest می‌پردازد.

### Hamcrest matcher framework 26-هدف از

یک فریم ورک و مجموعه ای از کتابخانه های اجرای تست های نرم افزاری هست. فریم ورک مجبور به شما این امکان را می دهد تا به واسطه‌ی کلاس های matcher از پیش موجود، شرایط خاصی را بررسی کرده و از صحت آن ها اطمینان حاصل نمایید. علاوه بر آن، شما می توانید matcher های خود را با پیاده سازی اختصاصی تعییه نمایید.

نسل سوم از فریم ورک matcher هست. اولین نسل از این فریم ورک از assert(logical statement) استفاده می کرد که کاستی قابل توجه آن عدم خوانایی بالا بود. نسل دوم توابع خاصی برای assertion و انتظار دریافت خروجی مورد نظر همچون assertEquals() ارائه دادند. این رویکرد سبب اضافه شدن توابع assert متعدد دیگری به Hamcrest که نسل سوم از این فریم ورک می باشد، شد. assertThat از متدهای نام assertThat و یک عبارت matcher برای بررسی اینکه تست موفق آمیز بود یا خیر استفاده می کند.

سعی دارد تا تست ها را تا حد امکان خوانا، مختصر و کاربر پسند تعییه نماید. به طور مثال، متدهای `is` یک تابع میزبان یا wrapper برای equalTo(value) می باشد.

```
import static org.hamcrest.MatcherAssert.assertThat;
import static org.hamcrest.Matchers.is;
import static org.hamcrest.Matchers.equalTo;
boolean a;
boolean b;
// all statements test the same
assertThat(a, equalTo(b));
assertThat(a, is(equalTo(b)));
assertThat(a, is(b));
```

تکه کدهای زیر دستورات assert خالص matcher های Hamcrest را با مقایسه می کنند.

```
// JUnit 4 for equals check
assertEquals(expected, actual);
// Hamcrest for equals check
assertThat(actual, is(equalTo(expected)));
// JUnit 4 for not equals check
assertFalse(expected.equals(actual));
// Hamcrest for not equals check
assertThat(actual, is(not(equalTo(expected))));
```

می توانید matcher ها را با استفاده از `anyOf` یا `allof` به صورت زنجیره ای و پشت سرهم فراخوانی نمایید.

```
assertThat("test", anyOf(is("testing"), containsString("est")));
```

در کل، پیغام های خطایی که Hamcrest تولید می کند، بسیار خوانا می باشند. علاوه بر آن در استفاده از Hamcrest، ویژگی type safety زبان جاوا به خوبی رعایت شده و matcher ها که از انواع داده ای generic استفاده می کنند دیگر با مشکل برخواهند خورد (کد type-safe اجازه ای دسترسی به بخش های نامربوط حافظه را نمی دهد. به عبارت دیگر تنها به آن بخش هایی از حافظه که امکان پذیر است اجازه ای دسترسی را می دهد).

## 27-7-استفاده از matcher ها در Hamcrest

### 27-7-1-اعلان dependency های مربوطه برای Gradle

جهت استفاده از Hamcrest matcher ها در پروژه ای که برای کامپایل از سیستم Gradle استفاده می کند، کافی است کتابخانه / dependency های زیر را به آن اضافه نمایید.

```
dependencies {
    // Unit testing dependencies
    testCompile 'junit:junit:4.12'
    // Set this dependency if you want to use Hamcrest matching
    testCompile 'org.hamcrest:hamcrest-library:1.3'
}
```

### 27-7-2-اعلان dependency های لازم برای سیستم Maven

جهت استفاده از Hamcrest در پروژه ای که برای کامپایل از سیستم Maven استفاده می کند، لازم است dependency زیر را به فایل pom خود اضافه نمایید.

```
<dependency>
<groupId>org.hamcrest</groupId>
<artifactId>hamcrest-library</artifactId>
<version>1.3</version>
```

```
<scope>test</scope>
</dependency>
```

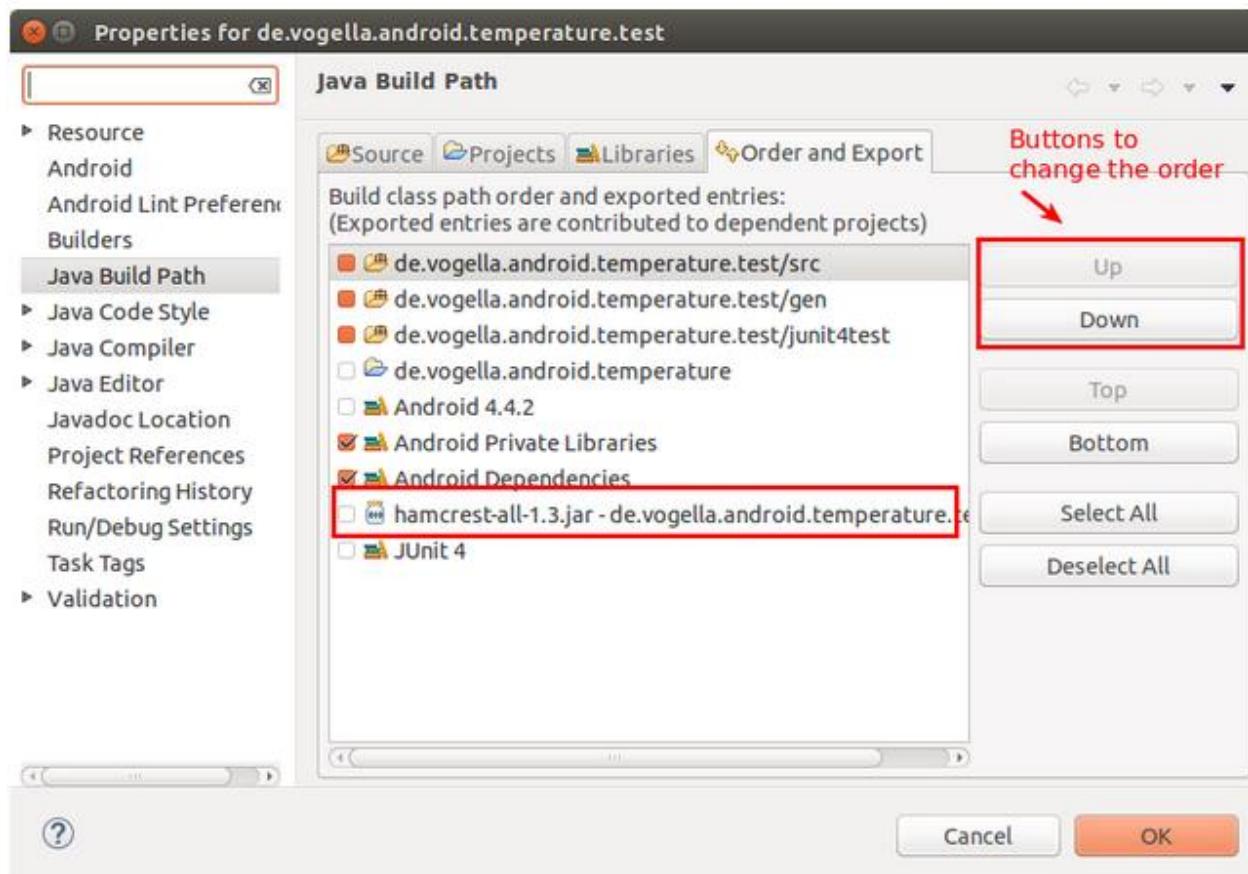
### 3-27-7- اضافه کردن Hamcrest به طور مستقیم به classpath پروژه در

#### Eclipse محیط کاری

ویرایشی از JUnit که در محیط برنامه نویسی Eclipse نصب می شود، صرفا های matcher اصلی Hamcrest را شامل می شود. جهت استفاده از تمامی matcher های موجود، آخرین نسخه ی hamcrest-all-\* jar را از آدرس

<https://code.google.com/p/hamcrest/downloads/list> دانلود نمایید و سپس آن را به classpath پروژه ی خود اضافه کنید.

**نکته:** در صورت بروز خطا با "org.hamcrest.Matchers"s signer information does not match signer information of hamcrest jar که فایل hamcrest jar قبل از اطمینان حاصل نمایید، اطمینان حاصل نمایید که other classes in the same package" قرار گرفته است. جهت ویرایش ترتیب قرارگیری می توانید به Order project properties در Eclipse تحت آدرس Java Build Path در تب and Export مراجعه نمایید.



## ۷-۲۸-۱ استفاده از Hamcrest matcher از list برای ها

کاربرد عملی از Hamcrest matcher های در کد زیر به نمایش گذاشته شده است.

```
assertThat(Long.valueOf(1), instanceof(Integer.class));
```

## ۷-۲۸-۲ استفاده از Hamcrest matcher از list برای ها

استفاده از Hamcrest matcher های list برای نوع داده ای در تکه کد زیر به نمایش گذاشته شده است.

```

import org.junit.Test;
import java.util.Arrays;
import java.util.List;
import static org.hamcrest.MatcherAssert.assertThat;
import static org.hamcrest.Matchers.contains;
import static org.hamcrest.Matchers.containsInAnyOrder;
import static org.hamcrest.Matchers.greaterThan;
import static org.hamcrest.collection.IsCollectionWithSize.hasSize;
import static org.hamcrest.core.Every.everyItem;
public class HamcrestListMatcherExamples {

    @Test
    public void listShouldInitiallyBeEmpty() {
        List<Integer> list = Arrays.asList(5, 2, 4);
        assertThat(list, hasSize(3));
        // ensure the order is correct
        assertThat(list, contains(5, 2, 4));
        assertThat(list, containsInAnyOrder(2, 4, 5));
        assertThat(list, everyItem(greaterThan(1)));
    }
}
// Check that a list of objects has a property race and
// that the value is not ORC
assertThat(fellowship, everyItem(hasProperty("race", is(not((ORC))))));

```

## 28-7-2- مرور کلی بر Hamcrest matcher های

مهم ترین matcher های فریم ورک مزبور در زیر شرح داده شده اند:

- - چنانچه تمامی matcher ها با یکدیگر مطابقت داشته باشند، همخوانی رخ می دهد.
- - چنانچه هر کدام از matcher ها مطابقت داشته باشد، همخوانی رخ می دهد.
- - در صورتی که گنجانده شده در wrapper مطابقت نداشته باشد و بالعکس، همخوانی رخ می دهد.
- - با استفاده از متدهای equals بررسی می کند آیا آبجکت ها با هم برابر هستند یا خیر.
- - یک پیشوند برای equalTo است که صرفا جهت بهبود خوانایی کد به این متده اضافه می شود.
- - Object.toString - hasToString

- – نوع را تست می کند.
- – بررسی می کند آیا مقدار null است یا خیر.
- sameInstance – هویت آبجکت را بررسی می کند. در واقع بررسی می کند آیا این آبجکت از همان نمونه است یا خیر.
- – بررسی می کند آیا نوع داده ای map آیتمی داخل خود دارد یا خیر.
- hasItems, hasItem – بررسی می کند آیا یک collection داخل خود آیتم هایی را دارد یا تهی است.
- hasItemInArray – بررسی می کند آیا یک آرایه تهی است یا حاوی آیتم و مقداری می باشد.
- closeTo – بررسی می کند آیا مقادیر ممیز شناور نزدیک به مقدار مورد نظر هستند یا خیر.
- lessThanOrEqualTo, lessThan, greaterThanOrEqualTo, greaterThan – بررسی می کند آیا رشته ها با هم برابر هستند یا خیر.
- – لازم به ذکر است که در تست فضای خالی نادیده گرفته می شود.
- startsWith, endsWith, containsString – بررسی می کند آیا رشته ها با هم مطابقت دارند یا خیر.

## 29-7- ساخت Hamcrest matcher اختصاصی خود

توسعه دهنده می تواند اعضای کلاسی به نام TypeSafeMatcher را به ارث برده (extend) و بدین وسیله Hamcrest matcher اختصاصی خود را بنویسد. مثال زیر یک نمونه از پیاده سازی matcher را نمایش می دهد که وظیفه‌ی آن بررسی تطابق بین یک String با عبارت باقاعده‌ی (regular expression) مورد نظر می باشد.

```
import org.hamcrest.Description;
import org.hamcrest.TypeSafeMatcher;
public class RegexMatcher extends TypeSafeMatcher<String> {
```

```

private final String regex;
public RegexMatcher(final String regex) {
    this.regex = regex;
}
@Override
public void describeTo(final Description description) {
    description.appendText("matches regular expression=\"" + regex + "\"");
}

@Override
public boolean matchesSafely(final String string) {
    return string.matches(regex);
}
// matcher method you can call on this matcher class
public static RegexMatcher matchesRegex(final String regex) {
    return new RegexMatcher(regex);
}
}

```

نمونه کد زیر نحوه‌ی استفاده و پیاده‌سازی آن را نمایش می‌دهد.

```

package com.vogella.android.testing.applicationtest;
import org.junit.Test;
import static org.hamcrest.MatcherAssert.assertThat;
public class TestCustomMatcher {
    @Test
    public void testRegularExpressionMatcher() throws Exception {
        String s = "aaabbbaaaa";
        assertThat(s, RegexMatcher.matchesRegex("a*b*a*"));
    }
}

```

## بخش پنجم :

# نوشتن تست نرم افزاری برای اپلیکیشن با فریم ورک

## AssertJ

در مبحث حاضر نحوه‌ی نوشتن تست‌های نرم افزاری مختصر، خوانا و کارامد برای اپلیکیشن‌های اندرویدی خود با استفاده از فریم‌ورک AssertJ را خواهید آموخت.

### 30-7-AssertJ مقدمه‌ای بر

AssertJ نیز یک کتابخانه اجرای تست نرم افزاری همچون Hamcrest است که با ارائه‌ی تعداد زیاری دستور assert و interface کارا به توسعه دهنده این اجازه را می‌دهد تا تست‌های نگهداشت پذیر و کارامد برای پروژه‌ی اندرویدی خود بنویسند. در واقع AssertJ یک شاخه از کتابخانه‌ای از دستورات assert به نام Fest است که امروزه دیگر به صورت حرفه‌ای پشتیبانی نمی‌شود.

کتابخانه‌ی مزبور بیشتر جهت آسان سازی نوشتن دستورات assert در تست‌های نرم افزاری و بهبود خوانایی آن‌ها تعییه شد. این کتابخانه با برخورداری از interface کارامد، به توسعه دهنده امکان می‌دهد تا دستورات assert را به صورت کارا و روان (پشت سرهم) جهت تست بخش‌های نرم افزار بنویسد.

متدهای پایه و اصلی کتابخانه‌ی نام برده assertThat بوده و دستورات مربوط به assert (بررسی صحت اجرای دستور و برآورده شدن انتظارات) بلافاصله پس از این متدهای درج می‌شوند.

## 31-7-استفاده از Assert لایه ای کاربردی

### Gradle-7-32

جهت استفاده از Assert همراه با سیستم کامپایل Gradle در پروژه، لازم است کتابخانه زیر را به فایل Gradle اضافه نمایید.

```
testCompile 'org.assertj:assertj-core:2.0.0'
```

### Maven-7-31-2

به منظور استفاده از کتابخانه ای مورد نظر در پروژه ای که با سیستم Maven کامپایل می شود، کافی است dependency (کتابخانه ای) زیر را به فایل pom خود اضافه نمایید.

```
<dependency>
<groupId>org.assertj</groupId>
<artifactId>assertj-core</artifactId>
<!-- use 3.0.0 for Java 8 projects -->
<version>2.0.0</version>
<scope>test</scope>
</dependency>
```

### 31-7- تنظیمات مرتبط با محیط کاری Eclipse

جهت استفاده از متده assertThat از کتابخانه ای Assert، پس از طی کردن مسیر رو به رو:   
Window > Preferences > Java > Editor > Content assist > Favorites > New Type دستور org.assertj.core.api.Assertions را درج نمایید.

پس از درج دستور فوق بایستی عبارت org.assertj.core.api.Assertions.\* را در کد مشاهده نمایید.

### 31-7- تنظیمات مرتبط با محیط کاری IntelliJ

برای استفاده از محیط برنامه نویسی IntelliJ نیاز به تنظیمات خاصی نیست. کافی است واژه assertThat را تایپ نموده و سپس با فشردن کلیدهای (Ctrl-Space)، قابلیت تکمیل خودکار کد (code completion) محیط کاری میزبان را دو بار فراخوانی نمایید.

## مثالی کاربردی از AssertJ

نمونه کد زیر از صفحه ای اصلی سایت AssertJ اقتباس شده است.

```
// unique entry point to get access to all assertThat methods and utility methods (e.g. entry)
import static org.assertj.core.api.Assertions.*;
// common assertions
assertThat(frodo.getName()).isEqualTo("Frodo");
assertThat(frodo).isNotEqualTo(sauron)
    .isIn(fellowshipOfTheRing);
// String specific assertions
assertThat(frodo.getName()).startsWith("Fro")
    .endsWith("do")
    .isEqualIgnoringCase("frodo");
// collection specific assertions
assertThat(fellowshipOfTheRing).hasSize(9)
    .contains(frodo, sam)
    .doesNotContain(sauron);
// using extracting magical feature to check fellowshipOfTheRing characters name :)
assertThat(fellowshipOfTheRing)
    .extracting("name")
    .contains("Boromir", "Gandalf", "Frodo", "Legolas")
    .doesNotContain("Sauron", "Elrond");
// map specific assertions, ringBearers initialized with the elves rings and the one ring bearers.
assertThat(ringBearers).hasSize(4)
    .contains(entry(oneRing, frodo), entry(nenya, galadriel))
    .doesNotContainEntry(oneRing, aragorn);
// and many more assertions : dates, file, numbers, exceptions ...
```

: بخش ششم :

## تست لایه ای UI اپلیکیشن های اندرویدی به واسطه ای Espresso ورک

726

آدرس آموزشگاه : تهران - خیابان شریعتی - بالاتر از خیابان ملک - جنب بانک صادرات - پلاک 651 طبقه دوم - واحد 7

88146323 - 88446780 - 88146330

آموزش حاضر به شرح نحوه‌ی توسعه‌ی تست‌های نرم افزاری برای لایه‌ی رابط کاربری اپلیکیشن‌های اندرویدی با استفاده از فریم‌ورک تست‌گیری Espresso می‌پردازد. جهت درک کامل مفاهیم این مبحث می‌بایست با اصول برنامه نویسی با زبان Java و چارچوب کاری Android آشنا باشید.

## 7-32-فریم‌ورک تست‌گیری Espresso و تست لایه‌ی UI اپلیکیشن

یک فریم‌ورک جهت آسان سازی تعییه‌تست‌های نرم افزاری قابل اطمینان برای لایه‌ی رابط کاربری پروژه‌های اندرویدی می‌باشد. شرکت گوگل این فریم‌ورک را برای اولین بار در اکتبر سال 2013 معرفی کرد.

از ویرایش 2.0 بخشی از Android Support Repository Espresso محسوب می‌شود. این فریم‌ورک تست‌گیری به صورت خودکار عملیات تست‌گیری شما را با لایه‌ی UI اپلیکیشن هماهنگ می‌سازد.

اطمینان حاصل می‌کند که قبل از به اجرا در آمدن تست، activity مورد نظر حتماً راه اندازی شده باشد. از دیگر ویژگی‌های جالب توجه فریم‌ورک مذکور این است که منتظر می‌ماند تا تمامی background activity‌ها پایان یابند و سپس خود به اجرا در می‌آید.

در اصل برای تست قابلیت‌های یک اپلیکیشن (اطمینان از عملکرد صحیح اپلیکیشن در سطح UI یک اپلیکیشن) در آن واحد طراحی شده، اما می‌توان از آن برای تست تعامل اپلیکیشن‌ها با هم نیز بهره‌گرفت. چنانچه قصد تست تعامل بین اپلیکیشن‌ها را دارید (نه صرفاً بررسی صحت عملیاتی که در سطح UI یک اپلیکیشن رخ می‌دهد)، لازم است از تکنیک تست‌گیری black box استفاده نمایید (تکنیک تست‌گیری black box به روی در تست نرم‌افزار اشاره دارد که در آن فرض می‌شود اطلاعاتی در مورد جزئیات داخلی اپلیکیشن مورد نظر وجود ندارد و تمرکز تست‌ها منحصر است بر روی خروجی‌های مختلف در مقابل ورودی‌های متفاوت است).

در کل از سه کامپوننت نرم افزاری زیر تشکیل شده است:

- به شما امکان می دهد تا view مورد نظر را در نمودار درختی view ها view Matchers بازدید کنید.
- اجازه می دهد تا عملیاتی را بر روی view ها انجام دهید.
- به توسعه دهنده این امکان را می دهد تا جهت اطمینان از صحت اطلاعات مربوط به وضعیت view state / view مورد نظر (برآورده شدن انتظار خاصی در رابطه با وضعیت view)، view را تست کند.

## یک تست ساده‌ی Espresso

ساختار کلی تست های Espresso به صورت زیر می باشد:

```
onView(ViewMatcher)
    .perform(ViewAction)
    .check(ViewAssertion);
```

این تست : 1) view را پیدا می کند 2) عملیاتی بر روی view اجرا می نماید 3) با بررسی وضعیت view پی می برد آیا اطلاعات و وضعیت view با آنچه انتظار می رفت مطابقت دارد یا خیر.  
کد زیر استفاده‌ی کاربردی از Espresso را به نمایش می دهد.

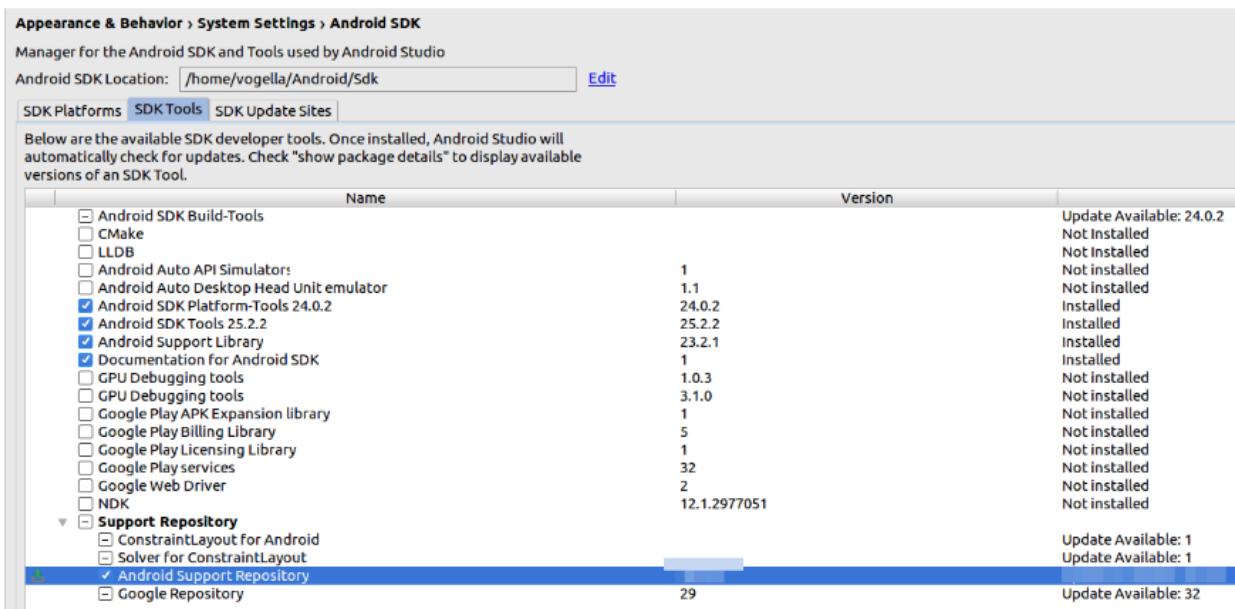
```
import static android.support.test.espresso.Espresso.onView;
import static android.support.test.espresso.action.ViewActions.click;
import static android.support.test.espresso.assertion.ViewAssertions.matches;
import static android.support.test.espresso.matcher.ViewMatchers.isDisplayed;
import static android.support.test.espresso.matcher.ViewMatchers.withId;
// image more code here...
// test statement
onView(withId(R.id.my_view)) // withId(R.id.my_view) is a ViewMatcher
    .perform(click()) // click() is a ViewAction
    .check(matches(isDisplayed())); // matches(isDisplayed()) is a ViewAssertion
// new test
onView(withId(R.id.greet_button))
    .perform(click())
    .check(matches(not(isEnabled())));
```

چنانچه Espresso با فراخوانی ViewMatcher قادر به یافتن view مورد نظر نبود، در آن صورت کل ساختار درختی view را در پیغام خطأ می گنجاند. این امر برای تجزیه و تحلیل و برطرف نمودن مشکل مفید می باشد.

## 33-7-نصب و استفاده از Espresso

### 33-7-1-نصب

ابتدا Android SDK manager را از طریق Android Support Repository نصب نمایید.



### 33-7-2-تنظیم فایل Gradle build برای استفاده از توابع Espresso

جهت استفاده از امکانات Espresso در تست های نرم افزاری خود، کتابخانه‌ی زیر را به فایل Gradle build اپلیکیشن اضافه نمایید.

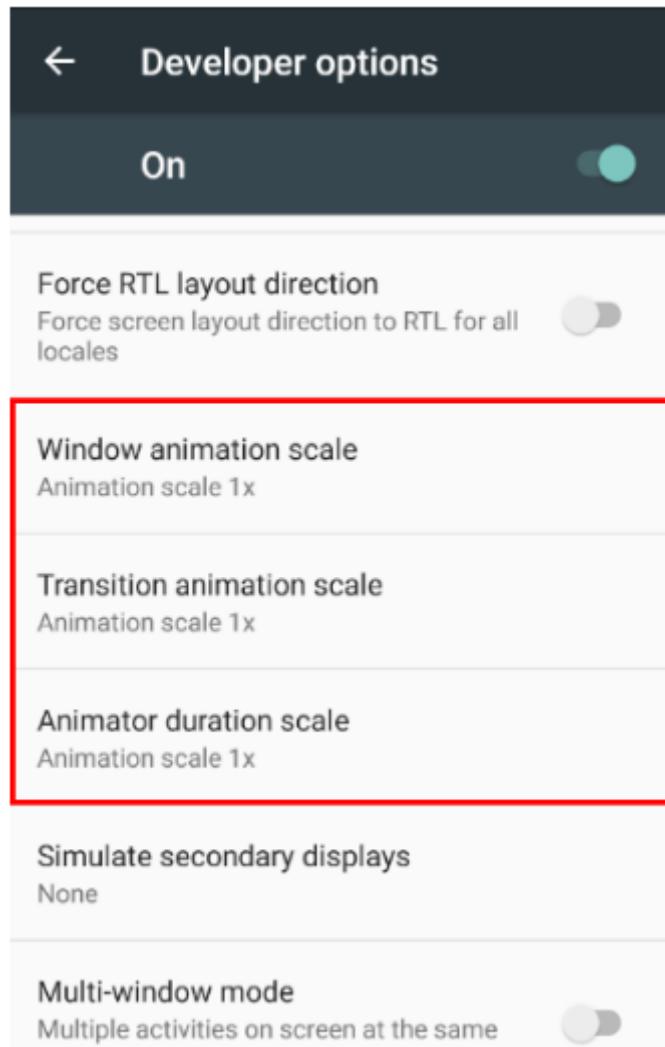
```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    testCompile 'junit:junit:4.12'
    // Android runner and rules support
    androidTestCompile 'com.android.support.test:runner:0.5'
    androidTestCompile 'com.android.support.test:rules:0.5'
    // Espresso support
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })
    // add this for intent mocking support
    androidTestCompile 'com.android.support.test.espresso:espresso-intents:2.2.2'
    // add this for webview testing support
    androidTestCompile 'com.android.support.test.espresso:espresso-web:2.2.2'
}
```

لازم است android.support.test.runner.AndroidJUnitRunner را به عنوان مقدار پارامتر build داخل فایل testInstrumentationRunner اپلیکیشن خود ذکر نمایید. شاید لازم باشد با توجه به کتابخانه‌ی موردنیاز استفاده، مقدار packagingOptions را از LICENSE.txt حذف نمایید. کد زیر مثالی در این زمینه را نشان می‌دهد.

```
apply plugin: 'com.android.application'
android {
    compileSdkVersion 22
    buildToolsVersion '22.0.1'
    defaultConfig {
        applicationId "com.example.android.testing.espresso.BasicSample"
        minSdkVersion 10
        targetSdkVersion 22
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    packagingOptions {
        exclude 'LICENSE.txt'
    }
    lintOptions {
        abortOnError false
    }
}
dependencies {
    // as before.....
}
```

### 7-33-3- تنظیمات دستگاه (Device settings)

اگر می‌خواهید از Espresso جهت تست لایه‌ی UI اپلیکیشن استفاده کنید، در آن صورت توصیه می‌شود اینیمیشن‌های سیستمی را غیرفعال نموده، بدین وسیله از رخداد خطا یا از کار افتادگی ناگهانی نرم افزار جلوگیری کنید و نیز مطمئن شوید که خروجی تست ثابت و تکرار پذیر می‌باشد.



## تمرین: استفاده ای کاربردی از Espresso جهت تست پروژه

### ایجاد پروژه ای و تست آن توسط Espresso

یک پروژه ای جدید اندرویدی به نام Espresso First با اسم پکیج Blank Espresso First ایجاد نمایید. این پروژه را بر اساس قالب آماده ای com.vogella.android.espressofirst Template ایجاد نمایید.

محتوای فایل layout پروژه ای خود، activity\_main.xml را به صورت زیر ویرایش نمایید.

```
<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:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">
    <EditText
        android:id="@+id/inputField"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <Button
        android:id="@+id/changeText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="New Button" android:onClick="onClick"/>
    <Button
        android:id="@+id/switchActivity"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Change Text" android:onClick="onClick"/>
</LinearLayout>
```

یک فایل جدید به نام activity\_second.xml ایجاد نمایید.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Large Text"
        android:id="@+id/resultView" />
</LinearLayout>
```

یک activity جدید با محتوای زیر ایجاد نمایید.

```
package com.vogella.android.espressofirst;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;
public class SecondActivity extends Activity{
    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```

super.onCreate(savedInstanceState);
setContentView(R.layout.activity_second);
TextView viewById = (TextView) findViewById(R.id.resultView);
Bundle inputData = getIntent().getExtras();
String input = inputData.getString("input");
viewById.setText(input);
}
}

```

سپس بدنہ ی کلاس `MainActivity` خود را نیز به صورت زیر ویرایش نمایید.

```

package com.vogella.android.espressofirst;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
public class MainActivity extends Activity {
    EditText editText;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        editText = (EditText) findViewById(R.id.inputField);
    }
    public void onClick(View view) {
        switch (view.getId()) {
            case R.id.changeText:
                editText.setText("Lalala");
                break;
            case R.id.switchActivity:
                Intent intent = new Intent(this, SecondActivity.class);
                intent.putExtra("input", editText.getText().toString());
                startActivity(intent);
                break;
        }
    }
}

```

### 7-33-4- تنظیم و ویرایش فایل `app build.gradle`

تنظیمات را طبق توضیحات مرتبیط با تنظیم فایل `Gradle build` برای استفاده از `Espresso` در مقاله ی حاضر انجام دهید.

### 7-33-5- تست نویسی برای پروژه بر اساس فریم ورک `Espresso`

```

package com.vogella.android.espressofirst;
import android.support.test.rule.ActivityTestRule;
import android.support.test.runner.AndroidJUnit4;

```

```

import org.junit.Rule;
import org.junit.Test;
import org.junit.runner.RunWith;
import static android.support.test.espresso.Espresso.onView;
import static android.support.test.espresso.action.ViewActions.click;
import static android.support.test.espresso.action.ViewActions.closeSoftKeyboard;
import static android.support.test.espresso.action.ViewActions.typeText;
import static android.support.test.espresso.assertion.ViewAssertions.matches;
import static android.support.test.espresso.matcher.ViewMatchers.withId;
import static android.support.test.espresso.matcher.ViewMatchers.withText;
@RunWith(AndroidJUnit4.class)
public class MainActivityEspressoTest {
    @Rule
    public ActivityTestRule<MainActivity> mActivityRule =
            new ActivityTestRule<>(MainActivity.class);
    @Test
    public void ensureTextChangesWork() {
        // Type text and then press the button.
        onView(withId(R.id.inputField))
                .perform(typeText("HELLO"), closeSoftKeyboard());
        onView(withId(R.id.changeText)).perform(click());
        // Check that the text was changed.
        onView(withId(R.id.inputField)).check(matches(withText("Lalala")));
    }
    @Test
    public void changeText_newActivity() {
        // Type text and then press the button.
        onView(withId(R.id.inputField)).perform(typeText("NewText"),
                closeSoftKeyboard());
        onView(withId(R.id.switchActivity)).perform(click());
        // This view is in a different Activity, no need to tell Espresso.
        onView(withId(R.id.resultView)).check(matches(withText("NewText")));
    }
}

```

## 6-33-7- ساخت و تنظیم intent) start intent (دوم) اجرا و راه اندازی

پس از ارسال مقدار false به عنوان پارامتر سوم به متده است `ActivityTestRule`، می توانید به ساخت و تنظیم آجکت `intent` برای فراخوانی و اجرای `activity` دوم بپردازید. این عملیات در نمونه کد زیر به نمایش گذاشته شده است.

```
package com.vogella.android.testing.espressosamples;
```

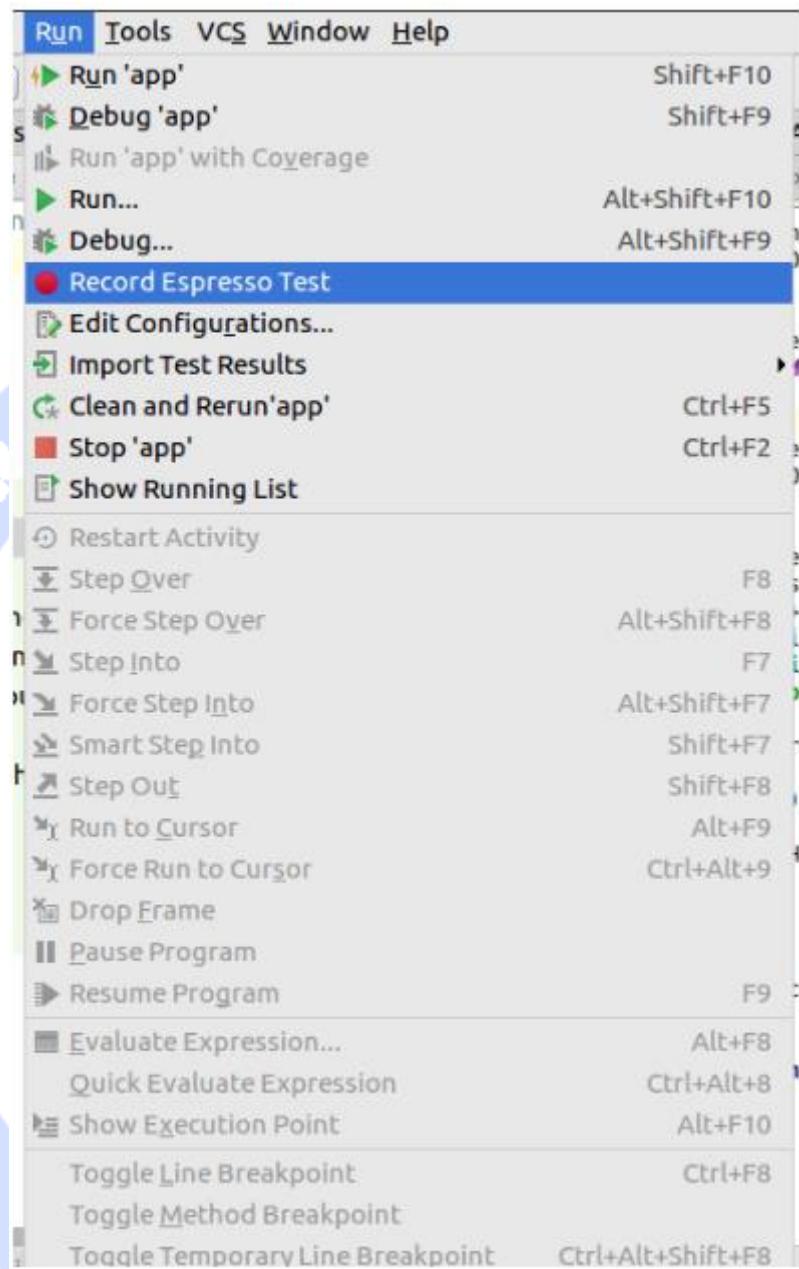
```

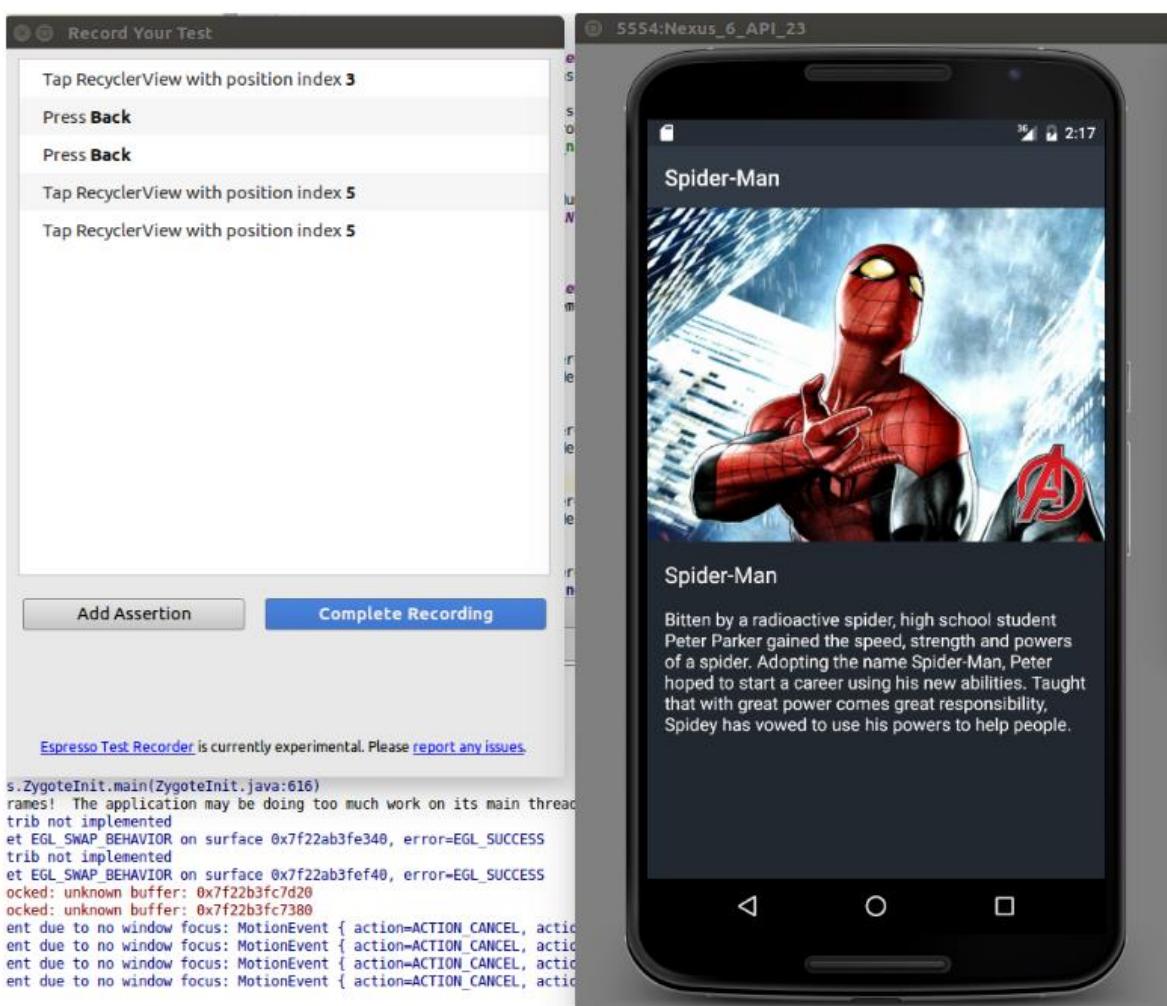
import android.content.Intent;
import android.support.test.rule.ActivityTestRule;
import android.support.test.runner.AndroidJUnit4;
import org.junit.Rule;
import org.junit.Test;
import org.junit.runner.RunWith;
import static android.support.test.espresso.Espresso.onView;
import static android.support.test.espresso.assertion.ViewAssertions.matches;
import static android.support.test.espresso.matcher.ViewMatchers.withId;
import static android.support.test.espresso.matcher.ViewMatchers.withText;
@RunWith(AndroidJUnit4.class)
public class SecondActivityTest {
    @Rule
    // third parameter is set to false which means the activity is not started automatically
    public ActivityTestRule<SecondActivity> rule =
        new ActivityTestRule(SecondActivity.class, true, false);
    @Test
    public void demonstrateIntentPrep() {
        Intent intent = new Intent();
        intent.putExtra("EXTRA", "Test");
        rule.launchActivity(intent);
        onView(withId(R.id.display)).check(matches(withText("Test")));
    }
}

```

### 7-33-7- ضبط تعامل با UI اپلیکیشن و جهت اجرای تست Espresso (Espresso UI recorder)

محیط برنامه نویسی Android Studio در منوی Run خود آیتمی به نام Record Espresso Test دارد (جهت دسترسی به آن این مسیر را طی نمایید: Run ▶ Record Espresso Test) که با فعال شدن، تعامل کاربر با UI اپلیکیشن را ضبط کرده و از روی آن یک تست Espresso ایجاد می‌کند.





### آنچه داده 7-33-8 تنظیم activity مورد تست

می توانید به آبجکت `activity` مورد تست دسترسی داشته و توابع دلخواه را بر روی آن فراخوانی نمایید. فرض کنید می خواهید یک متدهد یک متدهد را به صورت زیر بر روی نمونه ای از کلاس `activity` خود صدا بزنید.

```
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void configureMainActivity(String Uri) {
        // do something with this
    }
}
```

```

    }
}

```

می توانید متدهای زیر بر روی نمونه ای از کلاس `activity` را به صورت زیر بر روی نمونه ای از کلاس `MainActivity` خود فراخوانی نمایید.

```

package com.vogella.android.myapplication;
import android.support.test.rule.ActivityTestRule;
import android.support.test.runner.AndroidJUnit4;
import org.junit.Rule;
import org.junit.Test;
import org.junit.runner.RunWith;
@RunWith(AndroidJUnit4.class)
public class ExampleInstrumentedTest {
    @Rule
    public ActivityTestRule<MainActivity> mActivityRule =
            new ActivityTestRule<MainActivity>(MainActivity.class);
    @Test
    public void useApplicationContext() throws Exception {
        MainActivity activity = mActivityRule.getActivity();
        activity.configureMainActivity("http://www.vogella.com");
        // do more
    }
}

```

**نکته:** همچنین می توانید متدهای مورد نظر را در `ActivityTestRule` بازنویسی (`override`) نمایید. برای مثال، بدنه ای توابع `beforeActivityLaunched` و `afterActivityLaunched` را بازنویسی نمایید.

می توانید به `activity` جاری نیز دسترسی داشته باشید.

```

@Test
public void navigate() {
    Activity instance = getActivityInstance();
    onView(withText("Next")).perform(click());
    Activity activity = getActivityInstance();
    boolean b = (activity instanceof SecondActivity);
    assertTrue(b);
    // do more
}
public Activity getActivityInstance() {
    final Activity[] activity = new Activity[1];
    InstrumentationRegistry.getInstrumentation().runOnMainSync(new Runnable() {
        public void run() {
            Activity currentActivity = null;
            Collection resumedActivities =
                    ActivityLifecycleMonitorRegistry.getInstance().getActivitiesInStage(RESUMED);

```

(1)

```

        if (resumedActivities.iterator().hasNext()){
            currentActivity = (Activity) resumedActivities.iterator().next();
            activity[0] = currentActivity;
        }
    });
return activity[0];
}

```

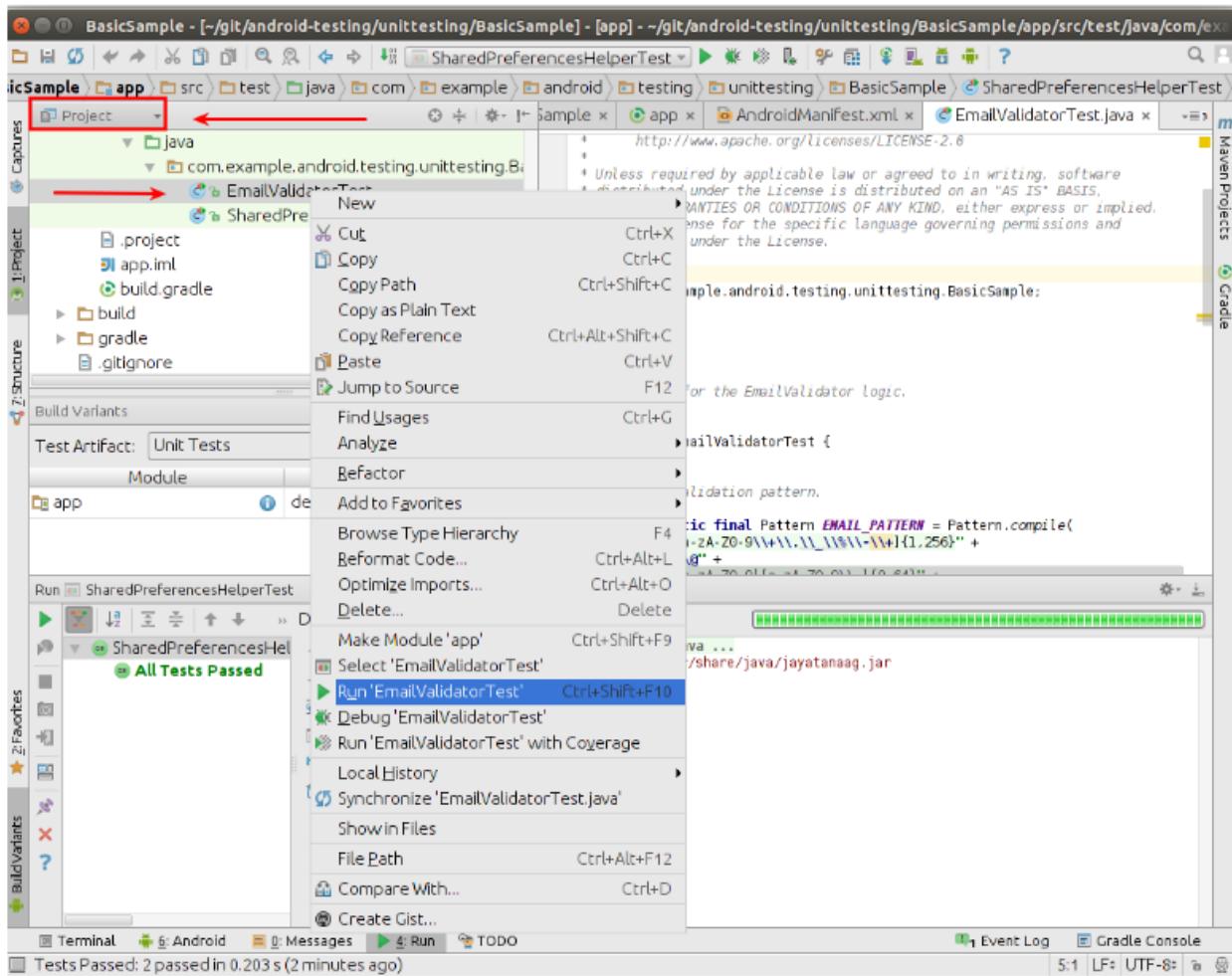
امکان دسترسی به activity جاری را فراهم می سازد.

**توجه:** لازم به ذکر است که دستور ActivityLifecycleMonitorRegistry جزء توابع کتابخانه ای / API نیست و به همین جهت امکان تغییر آن در آینده وجود دارد.

### 34-7-اجرای تست های Espresso

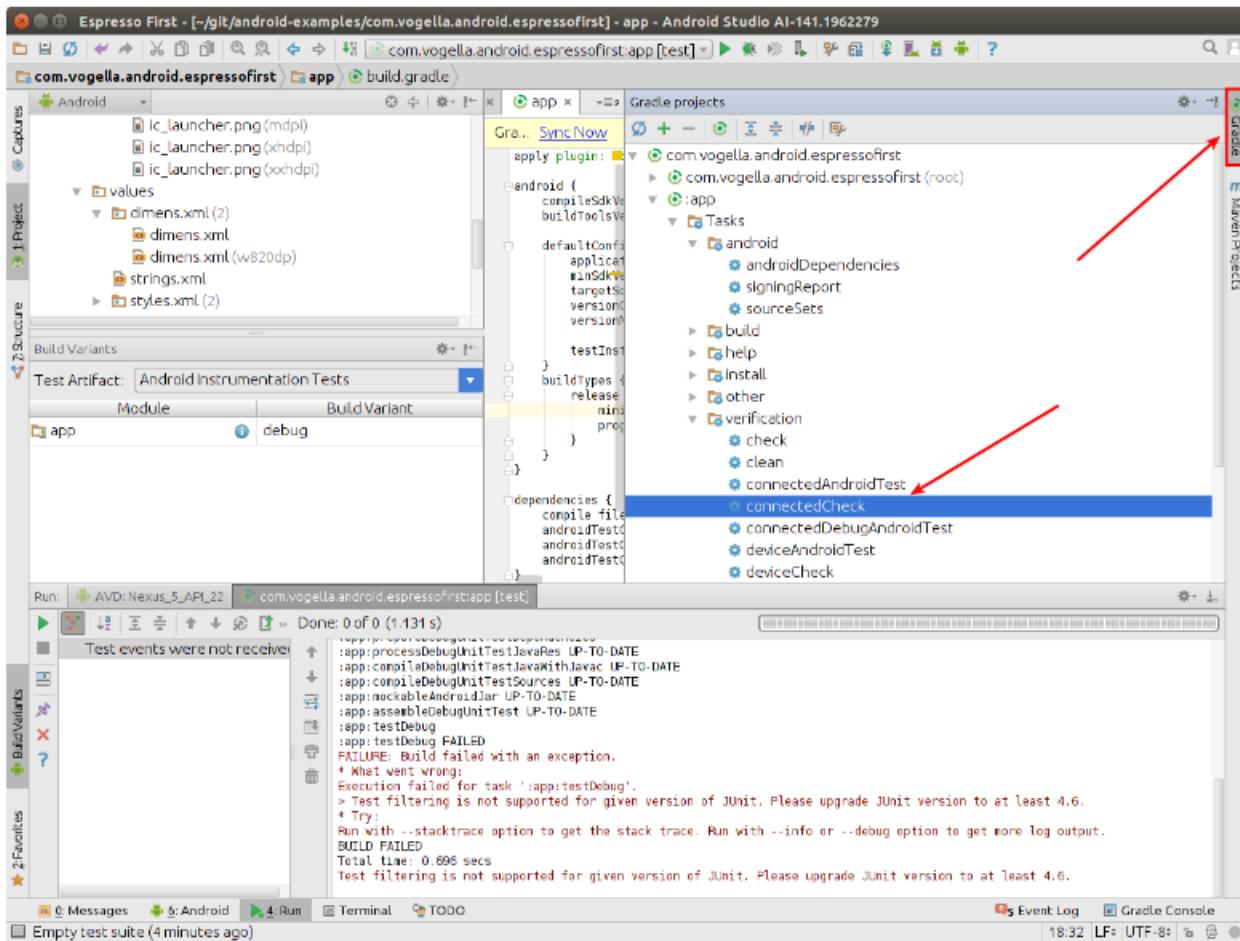
#### 34-1-اجرای تست در محیط کاری Android Studio

بر روی تست راست کلیک کرده و سپس گزینه **i Run** را انتخاب نمایید.



## 7-34-2- استفاده از سیستم کامپائل Gradle

بر روی آیکون کوچک Gradle در نوار کناری سمت راست محیط کلیک نموده و سپس گزینه `connectedCheck` را انتخاب نمایید تا تست به طور مستقیم از Gradle اجرا شود.



### 7-34-3-بررسی صحت نمایش پیغام toast

در زیر مثالی را مشاهده می کنید که در آن بررسی می شود آیا پس از کلیک بر روی آیتمی از لیست یک پیغام toast در نمایشگر ارائه می شود یا خیر.

```
package com.vogella.android.test.junTEXAMPLES;
import android.support.test.rule.ActivityTestRule;
import android.support.test.runner.AndroidJUnit4;
import org.junit.Rule;
import org.junit.Test;
import org.junit.runner.RunWith;
import static android.support.test.espresso.Espresso.onView;
import static android.support.test.espresso.Espresso.onView;
import static android.support.test.espresso.action.ViewActions.click;
import static android.support.test.espresso.assertion.ViewAssertions.matches;
import static android.support.test.espresso.matcher.RootMatchers.withDecorView;
import static android.support.test.espresso.matcher.ViewMatchers.isDisplayed;
```

```

import static android.support.test.espresso.matcher.ViewMatchers.withText;
import static org.hamcrest.CoreMatchers.containsString;
import static org.hamcrest.CoreMatchers.is;
import static org.hamcrest.CoreMatchers.startsWith;
import static org.hamcrest.Matchers.greaterThan;
import static org.hamcrest.Matchers.hasToString;
import static org.hamcrest.Matchers.instanceOf;
import static org.hamcrest.Matchers.not;
import static org.hamcrest.Matchers.notNullValue;
import static org.junit.Assert.assertThat;
@RunWith(AndroidJUnit4.class)
public class MainActivityTestList {
    @Rule
    public ActivityTestRule<MainActivity> rule = new ActivityTestRule<>(MainActivity.class);
    @Test
    public void ensureListViewsPresent() throws Exception {
        onData(hasToString(containsString("Frodo"))).perform(click());
        onView(withText(startsWith("Clicked:"))).
        inRoot(withDecorView(
            not(is(rule.getActivity().
            getWindow().getDecorView())))).
        check(matches(isDisplayed())));
    }
}

```

## 35- شبیه سازی آبجکت های Intent با توابع کتابخانه ای Espresso

این اجازه را به توسعه دهنده می دهد تا آبجکت های Intent را شبیه سازی / mock کند ( تعريف یک آبجکت intent ساختگی که رفتار آبجکت واقعی intent را شبیه سازی می کند). به واسطه ای این امکان توسعه دهنده قادر خواهد بود بررسی کند آیا کلاس activity آبجکت های intent مدعی نظر را صادر می کند یا خیر و نیز اینکه پس از دریافت خروجی مرتبط intent ، واکنش داخواه را نشان می دهد یا خیر.

فریم ورک تست گیری Espresso، این intent های ساختگی را در قالب کتابخانه ای به نام com.android.support.test.espresso:espresso-intents لازم است تنظیمات لازم را نیز در فایل Gradle build اعمال نمایید (جهت دریافت اطلاعات کامل به بخش مربوطه در مقاله ای حاضر مراجعه فرمایید).

اگر می خواهید از intent های ساختگی Espresso در تست های نرم افزاری خود استفاده کنید، لازم است بجای `IntentTestRule` ، از `ActivityTestRule` استفاده نمایید.

```
package testing.android.vogella.com.simpleactivity;
import android.support.test.espresso.intent.rule.IntentsTestRule;
import android.support.test.runner.AndroidJUnit4;
import org.junit.Rule;
import org.junit.Test;
import org.junit.runner.RunWith;
import static android.support.test.espresso.Espresso.onView;
import static android.support.test.espresso.action.ViewActions.click;
import static android.support.test.espresso.intent.Intents.intended;
import static android.support.test.espresso.intent.matcher.IntentMatchers.toPackage;
import static android.support.test.espresso.matcher.ViewMatchers.withId;
@RunWith(AndroidJUnit4.class)
public class TestIntent {
    @Rule
    public IntentsTestRule<MainActivity> mActivityRule =
            new IntentsTestRule<>(MainActivity.class);
    @Test
    public void triggerIntentTest() {
        onView(withId(R.id.button)).perform(click());
        intended(toPackage("testing.android.vogella.com.simpleactivity"));
    }
}
```

### 36-7-نوشتن تست برای بررسی عملکرد intent

#### 36-7-ایجاد پروژه‌ی آزمایشی (project under test)

یک پروژه‌ی جدید اندروید با اسم پکیج `testing.android.vogella.com.simpleactivity` بر اساس قالب آماده‌ی Empty Activity ایجاد نمایید.

حال با طی نمودن این مسیر: `File > New > Activity > Empty Activity`، یک activity جدید به نام `SecondActivity` به پروژه‌ی خود اضافه نمایید. این activity بایستی جهت تنظیم ظاهر خود یک فایل `layout` با حداقل یک المان `TextView` را فراخوانی نماید. شناسه (id) المان رابط کاربری `resultText` باید "resultText" و مقدار متنی آن بایستی بر روی "Started" تنظیم شده باشد.

یک فیلد `EditText` به فایل `layout` که ظاهر کلاس `MainActivity` را در UI تنظیم می‌کند، اضافه نمایید.

اکنون یک دکمه به فایل layout مورد استفاده ی MainActivity اضافه نمایید. پس از کلیک بر روی دکمه، دوم بایستی فراخوانی و اجرا شود.

فیلد EditText را به واسطه ی فراخوانی دستور intent.putExtra();، به عنوان کلید داخل آبجکت Intent قرار دهید. سپس String/مقدار رشته ای http://www.vogella.com را نیز با توجه به کلید "URL" و به وسیله ی پل ارتباطی extra (با ارسال آن به عنوان پارامتر اول به دستور intent.putExtra()) داخل آبجکت intent قرار دهید.

در زیر پیاده سازی نمونه از کلاس MainActivity را مشاهده می کنید.

```
package testing.android.vogella.com.simpleactivity;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void onClick(View view) {
        Intent intent = new Intent(this, SecondActivity.class);
        intent.putExtra("URL", "http://www.vogella.com");
        startActivity(intent);
    }
}
```

## نوشتمن تست

حال یک تست مبتنی بر فریم ورک Espresso بنویسید که موارد زیر را آزمایش کرده و از عملکرد صحیح اطمینان حاصل می نماید:

- بررسی کند آیا کلاس MainActivity یک المان دکمه با اسم ya شناسه ی R.id.button را دارد یا خیر.
- مطمئن شود مقدار نمایش داده شده بر روی دکمه رشته ی "Start new activity" می باشد.

- بررسی صحت فراخوانی متدهای `getActivity.onClick()` و `Intent extra (hasExtra("URL", "http://www.vogella.com"))` است یا آنکه آیا صحیح فراخوانی شده است یا خیر. این `Intent` بایستی دستور `extra (hasExtra("URL", "http://www.vogella.com"))` را در خود محصور داشته باشد.

## 7-36-2-بررسی صحت تست (اعتبارسنجی)

کد آزمایشی شما بایستی محتوایی مشابه زیر داشته باشد.

```
package testing.android.vogella.com.simpleactivity;
import android.support.test.espresso.intent.rule.IntentsTestRule;
import android.support.test.runner.AndroidJUnit4;
import org.junit.Rule;
import org.junit.Test;
import org.junit.runner.RunWith;
import static android.support.test.espresso.Espresso.onView;
import static android.support.test.espresso.action.ViewActions.click;
import static android.support.test.espresso.assertion.ViewAssertions.matches;
import static android.support.test.espresso.intent.Intents.intended;
import static android.support.test.espresso.intent.matcher.IntentMatchers.hasExtra;
import static android.support.test.espresso.intent.matcher.IntentMatchers.toPackage;
import static android.support.test.espresso.matcher.ViewMatchers.withId;
import static android.support.test.espresso.matcher.ViewMatchers.withText;
import static org.hamcrest.core.IsNull.notNullValue;
@RunWith(AndroidJUnit4.class)
public class TestIntent {
    @Rule
    public IntentsTestRule<MainActivity> mActivityRule = new IntentsTestRule<>(MainActivity.class);
    @Test
    public void triggerIntentTest() {
        // check that the button is there
        onView(withId(R.id.button)).check(matches(notNullValue()));
        onView(withId(R.id.button)).check(matches(withText("Start new activity")));
        onView(withId(R.id.button)).perform(click());
        intended(toPackage("testing.android.vogella.com.simpleactivity"));
        intended(hasExtra("URL", "http://www.vogella.com"));
    }
}
```

**تمرین: طراحی تست های functional (تست بخش های برنامه به صورت ایزوله) برای بررسی عملکرد صحیح activity**

## هدف از این تمرین

در تمرین حاضر، یک activity دیگر را فراخوانی کرده و اطمینان حاصل می کنید که مورد درستی اجرا می شود.

### 7-36-3- طراحی تست های functional برای activity ها

```
package testing.android.vogella.com.simpleactivity;
import android.support.test.rule.ActivityTestRule;
import android.support.test.runner.AndroidJUnit4;
import org.junit.Rule;
import org.junit.Test;
import org.junit.runner.RunWith;
import static android.support.test.espresso.Espresso.onView;
import static android.support.test.espresso.action.ViewActions.click;
import static android.support.test.espresso.assertion.ViewAssertions.matches;
import static android.support.test.espresso.matcher.ViewMatchers.withId;
import static android.support.test.espresso.matcher.ViewMatchers.withText;
@RunWith(AndroidJUnit4.class)
public class TestSecondActivityIsStarted {
    @Rule
    public ActivityTestRule<MainActivity> mActivityRule = new ActivityTestRule<>(MainActivity.class);
    @Test
    public void validateSecondActivity() {
        // check that the button is there
        onView(withId(R.id.button)).perform(click());
        onView(withId(R.id.resultText))
            .check(matches(withText("Started")));
    }
}
```

برای بررسی صحت اطمینان ویرایش مسقیم view، لازم است کلاس تست گیری زیر را برای آزمایش عملکرد کلاس SecondActivity ایجاد نمایید.

```
package testing.android.vogella.com.simpleactivity;
import android.support.test.rule.ActivityTestRule;
import android.support.test.runner.AndroidJUnit4;
import org.junit.Rule;
import org.junit.Test;
import org.junit.runner.RunWith;
import static android.support.test.espresso.Espresso.onView;
import static android.support.test.espresso.Espresso.pressBack;
```

```

import static android.support.test.espresso.action.ViewActions.click;
import static android.support.test.espresso.assertion.ViewAssertions.matches;
import static android.support.test.espresso.matcher.ViewMatchers.withId;
import static android.support.test.espresso.matcher.ViewMatchers.withText;
@RunWith(AndroidJUnit4.class)
public class SecondActivityFunctionalTest {
    @Rule
    public ActivityTestRule<MainActivity> mActivityRule = new ActivityTestRule<>(MainActivity.class);
    @Test
    public void validateSecondActivity() {
        // check that the button is there
        onView(withId(R.id.button)).perform(click());
        onView(withId(R.id.resultText))
            .check(matches(withText(("Started"))));
        pressBack();
        onView(withId(R.id.button))
            .check(matches(withText(("Start new activity"))));
    }
}

```

## آزمایش عملکرد کد ناهزمان با استفاده از فریم ورک Espresso گیری

تست عملکرد کد ناهزمان بدون کمک فریم ورک، امری بسیار پیچیده و ملال آور است. قبل از عرضه‌ی این فریم ورک، توسعه دهنده مجبور می‌شد مدت زمان مشخصی را جهت تست کد ناهزمان صبر نماید و یا به واسطه‌ی استفاده از نمونه‌ی کلاس CountDownLatch در test code (کد آزمایشی)، از پردازش ناهزمان اعلان نماید که پروسه به پایان رسیده است. از آنجایی که Espresso به صورت خودکار بر روی thread pool (انباره‌ای که thread‌ها در آن قرار دارند و تسك‌هایی که اغلب در قالب یک queue یا صف سازمان دهی شده اند را انجام می‌دهند) مربوط به AsyncTask نظارت دارد، این کار بسیار آسان می‌شود (کلاس AsyncTask یک کلاس abstract در چارچوب نرم افزاری اندروید است که تسك‌های زمان بر را در پس زمینه اجرا کرده و نتیجه را بدون اینکه بر روی thread اصلی تاثیر داشته باشد، در آن (UI thread) به صورت بهینه نمایش دهد). علاوه بر آن، Espresso صفتی که event‌های مربوط به UI اپلیکیشن در آن قرار گرفته اند را رصد کرده و سپس تنها زمانی اجرای تست را از سرمهی گیرد که هیچ تسكی در حال اجرا نباشد.

در صورت استفاده از منابع متفرقه همچون IntentService، بایستی اینترفیس IdlingResource را پیاده سازی نمایید. پیاده سازی interface مذکور بایستی بر این resource ناظارت داشته و آن را برای فریم ورک Espresso معرفی نماید (یک کلاس پایه است که برای پیاده سازی کامپوننت سرویس بایستی از آن ارث بری نمایید و وظیفه‌ی آن مدیریت درخواست‌های (ارائه شده در قالب آبجکت‌های intent) ناهمزمان می‌باشد).

ناشانگر یکی از منابع اپلیکیشن مورد تست است که امکانی را فراهم می‌کند تا همزمان با اجرای تست، عملیات پس زمینه‌ای ناهمزمان اجرا شود. برای مثال می‌توان به intent اشاره کرد که کلیک بر روی یک دکمه را پردازش می‌کند. به صورت پیش‌فرض، فریم ورک Espresso تمامی عملیات مربوط به view را علاوه بر UI Thread با AsyncTasks هماهنگ می‌سازد. اما، برای منابع hand-made هیچ راهی برای انجام این کار در اختیار ندارد. در چنین سناریوی، طراح تست می‌تواند منابع اختصاصی را به Espresso معرفی کند. پس از آن، Espresso صبر می‌کند تا resource مربوطه بی‌کار شود و بعد عملیات view را اجرا می‌نماید.

```
package com.vogella.android.espresso intentservice;
import android.app.ActivityManager;
import android.content.Context;
import android.support.test.espresso.IdlingResource;
import java.util.List;
public class IntentServiceIdlingResource implements IdlingResource {
    ResourceCallback resourceCallback;
    private Context context;
    public IntentServiceIdlingResource(Context context) {
        this.context = context;
    }
    @Override
    public String getName() {
        return IntentServiceIdlingResource.class.getName();
    }
    @Override
    public void registerIdleTransitionCallback(
            ResourceCallback resourceCallback) {
        this.resourceCallback = resourceCallback;
    }
    @Override
    public boolean isIdleNow() {
        boolean idle = !isIntentServiceRunning();
        if (idle && resourceCallback != null) {
            resourceCallback.onTransitionToIdle();
```

```

        }
        return idle;
    }

    private boolean isIntentServiceRunning() {
        ActivityManager manager =
            (ActivityManager) context.getSystemService(Context.ACTIVITY_SERVICE);
        // Get all running services
        List<ActivityManager.RunningServiceInfo> runningServices =
            manager.getRunningServices(Integer.MAX_VALUE);
        // check if our is running
        for (ActivityManager.RunningServiceInfo info : runningServices) {
            if (MyIntentService.class.getName().equals(
                info.service.getClassName())) {
                return true;
            }
        }
        return false;
    }

    package com.vogella.android.espresso intentservice;
    import android.app.Instrumentation;
    import android.content.Context;
    import android.support.test.InstrumentationRegistry;
    import android.support.test.espresso.Espresso;
    import android.support.test.rule.ActivityTestRule;
    import android.support.test.runner.AndroidJUnit4;
    import org.junit.After;
    import org.junit.Before;
    import org.junit.Rule;
    import org.junit.Test;
    import org.junit.runner.RunWith;
    import static android.support.test.espresso.Espresso.onView;
    import static android.support.test.espresso.action.ViewActions.click;
    import static android.support.test.espresso.assertion.ViewAssertions.matches;
    import static android.support.test.espresso.matcher.ViewMatchers.withId;
    import static android.support.test.espresso.matcher.ViewMatchers.withText;
    import static org.hamcrest.Matchers.notNullValue;
    @RunWith(AndroidJUnit4.class)
    public class IntegrationTest {
        @Rule
        public ActivityTestRule rule = new ActivityTestRule(MainActivity.class);
        IntentServiceIdlingResource idlingResource;
        @Before
        public void before() {
            Instrumentation instrumentation
                = InstrumentationRegistry.getInstrumentation();
            Context ctx = instrumentation.getTargetContext();
            idlingResource = new IntentServiceIdlingResource(ctx);
            Espresso.registerIdlingResources(idlingResource);
        }
        @After
        public void after() {
    
```

```

        Espresso.unregisterIdlingResources(idlingResource);
    }
    @Test
    public void runSequence() {
        // this triggers our intent service, as we registered
        // Espresso for it, Espresso wait for it to finish
        onView(withId(R.id.action_settings)).perform(click());
        onView(withText("Broadcast")).check(matches(notNullValue()));
    }
}

```

## تمرین: تست کاربردی کد ناهمزمان با استفاده از Espresso

یک پروژه جدید با اسم پکیج `testing.android.vogella.com.AsyncTask` ایجاد نمایید. پروژه حاضر باقیستی این امکان را فراهم کند تا با کلیک بر روی دکمه، یک نمونه از `AsyncTask` جهت مدیریت عملیات در پس زمینه به طور ناهمگام، فراخوانی شود.

نمونه ای از پیاده سازی `:activity`

```

package testing.android.vogella.com.AsyncTask;
import android.app.Activity;
import android.os.AsyncTask;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void onClick(View view) {
        TextView textView = (TextView) findViewById(R.id.text);
        textView.setText("Running");
        myTask.execute("test");
    }
    final AsyncTask<String, Void, String> myTask = new AsyncTask<String, Void, String>() {
        @Override
        protected String doInBackground(String... arg0) {
            return "Long running stuff";
        }
        @Override
        protected void onPostExecute(String result) {
            TextView textView = (TextView) findViewById(R.id.text);
            textView.setText("Done");
        }
    }
}

```

750

آدرس آموزشگاه : تهران - خیابان شریعتی - بالاتر از خیابان ملک - جنب بانک صادرات - پلاک 651 طبقه دوم - واحد 7

88146323 - 88446780 - 88146330

```
}
```

## کد پیاده سازی شده برای تست:

```
package testing.android.vogella.com.AsyncTask;
import android.content.Context;
import android.support.test.InstrumentationRegistry;
import android.support.test.rule.ActivityTestRule;
import android.support.test.runner.AndroidJUnit4;
import org.junit.Rule;
import org.junit.Test;
import org.junit.rules.TestName;
import org.junit.runner.RunWith;
import static android.support.test.espresso.Espresso.onView;
import static android.support.test.espresso.action.ViewActions.click;
import static android.support.test.espresso.assertion.ViewAssertions.matches;
import static android.support.test.espresso.matcher.ViewMatchers.withId;
import static android.support.test.espresso.matcher.ViewMatchers.withText;
@RunWith(AndroidJUnit4.class)
public class EspressoTest {
    @Rule
    public ActivityTestRule<MainActivity> mActivityRule =
        new ActivityTestRule<>(MainActivity.class);
    @Test
    public void buttonShouldUpdateText(){
        onView(withId(R.id.update)).perform(click());
        onView(withId(R.id.text)).check(matches(withText("Done")));
    }
}
```

## تمرین: تعریف matcher اختصاصی برای Espresso

matcher = یک موتور است که رشته ای از حروف را بر اساس الگو یا pattern تفسیر شده، بررسی کرده و سعی بر یافتن موارد منطبق دارد. matcher در واقع با فراخوانی متدهای pattern مورد نظر، ایجاد می شود.)

اندروید با ارائه ی کلاسی به نام BoundedMatcher به شما امکان می دهد تا های مورد نظر matcher) هایی بنویسید که با view مورد نظر منطبق باشد) را ویژه ی view های (جهت انطباق با view ای از نوع مورد نظر همچون button یا textbox) معین اعلان نماید.

حال بک matcher بنویسید که صحت عملکرد متن های راهنمای (text hint) فیلد EditText را بررسی می کند.

```
public static Matcher<View> withItemHint(String itemHintText) {
    checkArgument(!(itemHintText.equals(null)));
    return withItemHint(is(itemHintText));
}
public static Matcher<View> withItemHint(final Matcher<String> matcherText) {
    // use preconditions to fail fast when a test is creating an invalid matcher.
    checkNotNull(matcherText);
    return new BoundedMatcher<View, EditText>(EditText.class) {
        @Override
        public void describeTo(Description description) {
            description.appendText("with item hint: " + matcherText);
        }
        @Override
        protected boolean matchesSafely(EditText editTextField) {
            return matcherText.matches(editTextField.getHint().toString());
        }
    };
}
import static com.your.package.test.Matchers.withItemHint;
...
onView(withItemHint("test")).check(matches(isDisplayed()));
```

می توان از طریق کد زیر آن را انجام داد:

**بخش هفتم :**

## تست تعامل بین چندین کامپوننت نرم افزاری لایه ی UI اپلیکیشن با استفاده از فریم ورک UI Automator به

**روش black-box**

### 38- تست تعامل بین چندین کامپوننت نرم افزاری اپلیکیشن با استفاده از فریم ورک UI Automator

آموزش حاضر به شما می آموزد چگونه حتی برای اپلیکیشن هایی که source code آن ها را در دست ندارید، تست هایی به روش black-box طراحی کنید.

## 39-7- تست تعامل بین کامپوننت های نرم افزاری اپلیکیشن به روشن blackbox

### 39-1- استفاده از UI Automator جهت تست تعامل بین کامپوننت های اپلیکیشن (تست کل اپلیکیشن)

در مبحث حاضر تمامی کامپوننت های رابط کاربری اپلیکیشن به روش black box (functional) به روشن مورد تست قرار می گیرد.

SDK (مجموعه ابزار توسعه و طراحی اپلیکیشن) اندروید یک کتابخانه‌ی مبتنی بر جاوا به نام uiautomator دارد که به وسیله‌ی آن توسعه دهنده قادر خواهد بود تست نرم افزاری برای کامپوننت های UI اپلیکیشن نوشته و نیز با ارائه‌ی یک موتور امکان اجرای این تست‌ها را فراهم می‌آورد. لازم به ذکر است که برای استفاده از هر دو ابزار مزبور، بایستی ویرایش 4.3 اندروید (ورژن 18 توابع کتابخانه‌ای اندروید/API level) بر روی دستگاهی که تست بر روی آن اجرا می‌شود، نصب باشد.

کتابخانه `uiAutomator` با ارائه ی کلاس های `UiObject`, `UiDevice` و `UiSelector` که بر اساس کلاس `UiSelector` ساخته می شود، به ترتیب قابلیت های زیر را در اختیار توسعه دهنده قرار می دهد:

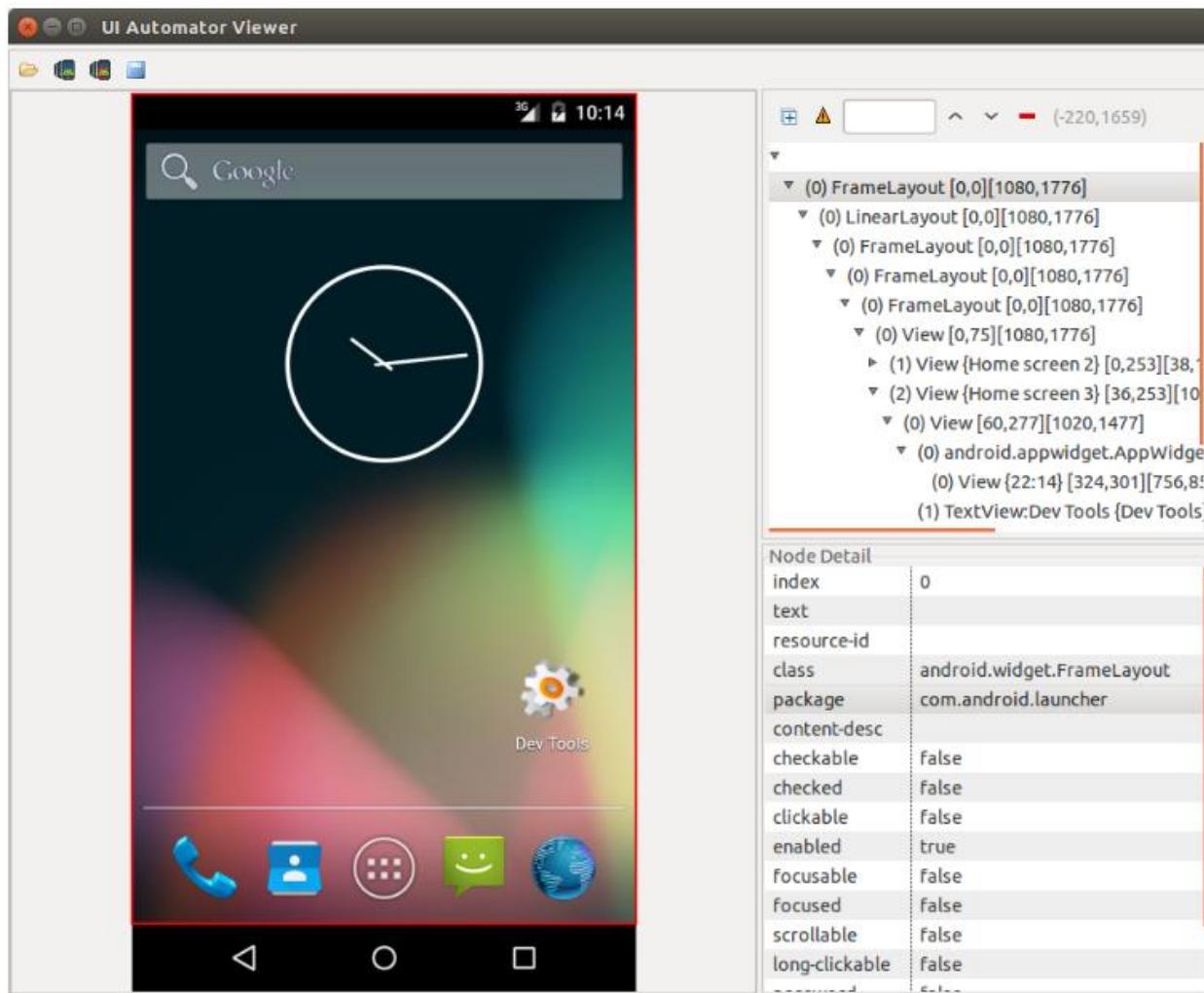
1. تبادل داده و تعامل با دستگاه میزبان
2. پیدا کردن المان ها در نمایشگر
3. ساخت آبجکت های تشکیل دهنده ی ظاهر برنامه و لایه ی UI

دو کلاس `UiCollection` و `UiScrollable` نیز به ترتیب این اجازه را می دهند تا چندین المان UI را همزمان انتخاب کرده و جهت دسترسی به المان مورد نظر در `view` پیمایش نماید.

## 7-39-2- دسترسی به اطلاعات مربوط به `view` با ابزار `uiAutomatorViewer`

چارچوب نرم افزاری اندروید ابزاری به نام `uiAutomatorViewer` را ارائه می دهد که به وسیله ی آن شما قادر خواهید بود لایه ی UI یک اپلیکیشن را به راحتی تجزیه و تحلیل کنید. شما می توانید با استفاده از این ابزار اندیس، متن یا attribute اپلیکیشن را پیدا کنید. در واقع این ابزار به شما امکان می دهد تا به نمودار درختی (layout hierarchy) المان های رابط کاربری دسترسی داشته و های مربوط به هریک از کنترل ها (المان های رابط کاربری) را مشاهده نمایید.

به واسطه ی این ابزار حتی افرادی که در نوشتن اپلیکیشن شرکت نداشته یا برنامه نویس نیستند می توانند به راحتی برنامه را تحلیل کرده و برای آن تست طراحی کنند. این را در نیز تصویر زیر مشاهده می کنید.



به منظور اجرای ابزار `uiautomatorviewer`, پس از ارائه ی آدرس/`android-sdk/tools/uiautomatorviewer` در خط فرمان، دستور `uiautomatorviewer` را درج نمایید.

## دسترسی و استفاده از UI Automator برای تست اپلیکیشن

### انجام تنظیمات لازم در پروژه

برای استفاده از ابزار UI Automator جهت اجرای تست بر روی اپلیکیشن، کتابخانه (`dependency`) کتابخانه (`dependency`) زیر را به فایل `build.gradle` پروژه اضافه نموده و `AndroidJUnitRunner` را به عنوان مقدار به

پارامتر `testInstrumentationRunner` پاس دهد. در زیر این مثال را به صورت کاربردی مشاهده می کنید.

```
apply plugin: 'com.android.application'
android {
    compileSdkVersion 19
    buildToolsVersion "23.0.0"
    defaultConfig {
        applicationId "com.vogella.android.test.uiautomator"
        minSdkVersion 19
        targetSdkVersion 23
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}
dependencies {
    androidTestCompile 'com.android.support.test:runner:0.3'
    androidTestCompile 'junit:junit:4.12'
    testCompile 'junit:junit:4.12'
    androidTestCompile 'org.hamcrest:hamcrest-library:1.3'
    androidTestCompile 'com.android.support.test:runner:0.3'
    androidTestCompile 'com.android.support.test:rules:0.3'
    androidTestCompile 'com.android.support.test.uiautomator:uiautomator-v18:2.1.1'
}
```

توصیه می شود جهت جلوگیری از بروز اختلال، این میشن را بر روی دستگاه اندرویدی که تست در بستر آن اجرا می شود، غیر فعال نمایید.

### 7-39-3- محل قرارگیری تست ها

تست های مبتنی بر Automator UI می بایست به صورت پیش فرض در پوشه `app/src/androidTest` جایگذاری شود.

### 7-39-4- نحوه نوشتن تست

تست های UI Automator می بایست با دستور `@RunWith(AndroidJUnit4.class)` نشانه `ActivityTestRule` خود را تنظیم نمایند.

### 7-39-5- ساخت پروژه و تنظیم فایل Gradle build

یک پروژه ای جدید که اسم پوشه ای بالای آن (top level project name) `com.example.android.testing.uiautomator` باشد، ایجاد نمایید.  
فایل `Gradle build` را به صورت زیر ویرایش نمایید.

```
apply plugin: 'com.android.application'
android {
    compileSdkVersion 19
    buildToolsVersion "23.0.0"
    defaultConfig {
        applicationId "com.vogella.android.test.uiautomator"
        minSdkVersion 19
        targetSdkVersion 23
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}
dependencies {
    androidTestCompile 'com.android.support.test:runner:0.3'
    androidTestCompile 'junit:junit:4.12'
    testCompile 'junit:junit:4.12'
    androidTestCompile 'org.hamcrest:hamcrest-library:1.3'
    androidTestCompile 'com.android.support.test:runner:0.3'
```

```

    androidTestCompile 'com.android.support.test:rules:0.3'
    androidTestCompile 'com.android.support.test.uiautomator:uiautomator-v18:2.1.1'
}

```

## ایجاد تست

تستی با پیاده سازی زیر را در پوشه `_androidTest` تعریف نمایید.

```

package com.vogella.android.test.uiautomator;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.content.pm.ResolveInfo;
import android.support.test.InstrumentationRegistry;
import android.support.test.rule.ActivityTestRule;
import android.support.test.runner.AndroidJUnit4;
import android.support.test.uiautomator.By;
import android.support.test.uiautomator.UiDevice;
import android.support.test.uiautomator.UiObject;
import android.support.test.uiautomator.UiObjectNotFoundException;
import android.support.test.uiautomator.UiScrollable;
import android.support.test.uiautomator.UiSelector;
import android.support.test.uiautomator.Until;
import org.hamcrest.Matchers;
import org.junit.Before;
import org.junit.Rule;
import org.junit.Test;
import org.junit.runner.RunWith;
import static org.hamcrest.Matchers.*;
import static org.junit.Assert.*;
@Rule
public class MyUiAutomatorTest {
    @Rule
    public ActivityTestRule<MainActivity> mActivityRule
        = new ActivityTestRule<MainActivity>(MainActivity.class);
    private UiDevice mDevice;
    @Before
    public void setUp() {
        // Initialize UiDevice instance
        mDevice = UiDevice.getInstance(InstrumentationRegistry.getInstrumentation());
        // Start from the home screen
        mDevice.pressHome();
        mDevice.wait(Until.hasObject(By.pkg(getLauncherPackageName()).depth(0)), 1000);
    }
    @Test
    public void checkSettings() throws UiObjectNotFoundException {
        // Simulate a short press on the HOME button.
        mDevice.pressHome();
        // We're now in the home screen. Next, we want to simulate
        // a user bringing up the All Apps screen.
        // If you use the uiautomatorviewer tool to capture a snapshot

```

```

// of the Home screen, notice that the All Apps button's
// content-description property has the value "Apps". We can
// use this property to create a UiSelector to find the button.
UiObject allAppsButton = mDevice.findObject(new UiSelector().description("Apps"));
// Simulate a click to bring up the All Apps screen.
allAppsButton.clickAndwaitForNewWindow();
// In the All Apps screen, the Settings app is located in
// the Apps tab. To simulate the user bringing up the Apps tab,
// we create a UiSelector to find a tab with the text
// label "Apps".
UiObject appsTab = mDevice.findObject(new UiSelector().text("Apps"));
// Simulate a click to enter the Apps tab.
appsTab.click();
// Next, in the apps tabs, we can simulate a user swiping until
// they come to the Settings app icon. Since the container view
// is scrollable, we can use a UiScrollable object.
UiScrollable appViews = new UiScrollable(
    new UiSelector().scrollable(true));
// Set the swiping mode to horizontal (the default is vertical)
appViews.setAsHorizontalList();
// create a UiSelector to find the Settings app and simulate
// a user click to launch the app.
UiObject settingsApp = appViews
    .getChildByText(new UiSelector()
        .className(android.widget.TextView.class.getName()),
        "Settings");
settingsApp.clickAndwaitForNewWindow();
// Validate that the package name is the expected one
UiObject settingsValidation = new UiObject(
    new UiSelector()
        .packageName("com.android.settings"));
assertThat(settingsValidation.exists(), equalTo(true));
}

private String getLauncherPackageName() {
    // Create launcher Intent
    final Intent intent = new Intent(Intent.ACTION_MAIN);
    intent.addCategory(Intent.CATEGORY_HOME);
    // Use PackageManager to get the launcher package name
    PackageManager pm = InstrumentationRegistry.getContext().getPackageManager();
    ResolveInfo resolveInfo = pm.resolveActivity(intent, PackageManager.MATCH_DEFAULT_ONLY);
    return resolveInfo.activityInfo.packageName;
}
}

```

## اجرا و بررسی صحت اجرای موققیت آمیز تست

تست را اجرا کرده و از عملکرد صحیح آن اطمینان حاصل نمایید.



بخش هشتم :

## تست لایه‌ی AI اپلیکیشن/رابط کاربری با استفاده از فریم ورک تست گیری Robotium

آموزش حاضر نحوه‌ی تست لایه‌ی AI اپلیکیشن/اجرای تست بر روی لایه‌ی AI اپلیکیشن اندرویدی را شرح می‌دهد.

## Robotium-7-40

یک افزونه برای فریم ورک تست گیری اندروید است و صرفا برای آسان سازی نوشتن تست های نرم افزاری برای UI اپلیکیشن طراحی شده است. تست های Robotium اعضای کلاس `test` هایی (مورد آزمایش) بنویسید که چندین `activity` را شامل می شود.

**نکته:** تست هایی که بر اساس Robotium طراحی می شوند منحصر با لایه UI اپلیکیشن تعامل دارند. به عبارت دیگر این نوع تست ها فرض می گیرند که اپلیکیشن مورد آزمایش یک جعبه سیاه است که محتویات و کد داخل آن اطلاعاتی ندارد.

همان طور که قبلا ذکر شد، نوعی افزونه برای فریم ورک تست گیری اندروید می باشد (چارچوب طراحی تست نرم افزاری اندروید را بسط می دهد) که نوشتن تست برای اپلیکیشن های اندرویدی را آسان می سازد.

کلاس اصلی فریم ورک مزبور برای طراحی تست، `Solo` می باشد. این کلاس به مجرد ساخت `test` و تست `activity` مقداردهی اولیه می شود.

## Robotium-41-7-نصب

جهت تست Robotium در پروژه‌ی اندرویدی خود، لازم است کتابخانه (dependency) مورد نیاز آخرين ويراييش Robotium را به فایل `build` اضافه نمایید.

```
dependencies {
    // Unit testing dependencies
    androidTestCompile 'com.jayway.android.robotium:robotium:5.4.12'
}
```

در زمان تنظیم مقاله‌ی حاضر، 5.4.1 جدیدترین ویرایش Robotium می باشد.

## 42-نمونه ای از پیاده سازی تست های Robotium

کد زیر نحوه ای پیاده سازی و استفاده از فریم ورک Robotium در تست یک activity (تست لایه ای رابط کاربری اپلیکیشن) را نمایش می دهد.

```
package de.vogella.android.test.target.test;
import junit.framework.Assert;
import android.test.ActivityInstrumentationTestCase2;
import com.robotium.solo.Solo;
import de.vogella.android.test.target.SimpleActivity;
import de.vogella.android.test.target.SimpleListActivity;
public class SimpleActivityTest extends
    ActivityInstrumentationTestCase2<SimpleActivity> {
    private Solo solo;
    public SimpleActivityTest() {
        super(SimpleActivity.class);
    }
    public void setUp() throws Exception {
        solo = new Solo(getInstrumentation(), getActivity());
    }
    @Override
    public void tearDown() throws Exception {
        solo.finishOpenedActivities();
    }
}
```

در زیر نمونه ای دیگری از کاربرد متدهای Robotium را مشاهده می کنید که در آن پیاده سازی یک لیست مورد آزمایش قرار می گیرد.

```
// check that we have the right activity
solo.assertCurrentActivity("wrong activity", SimpleActivity.class);
// Click a button which will start a new Activity
// Here we use the ID of the string to find the right button
solo.clickOnButton(solo.getString(R.string.button1));
// Validate that the Activity is the correct one
solo.assertCurrentActivity("wrong activity", SimpleListActivity.class);
solo.clickInList(1);
// searchForText has a timeout of 5 seconds
assertTrue(solo.waitForText("Android")); // Assertion
solo.clickInList(2);
assertTrue(solo.waitForText("iPhone")); // Assertion
solo.clickInList(3);
assertTrue(solo.waitForText("Blackberry")); // Assertion
```

```

solo.goBack();
solo.clickOnButton("Button2");
solo.clickOnButton("Button3");
// open the menu
solo.sendKey(Solo.MENU);
solo.clickOnText("Preferences");
solo.clickOnText("User");
solo.clearEditText(0);
Assert.assertTrue(solo.searchText(""));
solo.enterText(0, "http://www.vogella.com");
Assert.assertTrue(solo.searchText("http://www.vogella.com"));
solo.goBack();

```

## 43-توابع کتابخانه ای / Robotium API

Solo توابعی برای فراخوانی کامپوننت های رابط کاربری ( کامپوننت های UI اپلیکیشن اندرویدی ) در اختیار توسعه قرار می دهد. در جدول زیر تعدادی از این توابع را همراه با شرح کاربرد مشاهده می نمایید.

متدها	شرح
getView(int id)	بر اساس (پارامتر) ID ارائه شده، view مدنظر را در activity جاری پیدا کرده و به عنوان خروجی برمی گرداند.
waitForText(text)	انتظار دارد که activity جاری با دومین پارامتر ارسالی به تابع، برابر باشد.
clickOnText(text)	بر روی کنترل دکمه‌ی ای که مقدار متنی "text" را دارد کلیک می‌کند.
sendKey(Solo.MENU);	Sends the menu key event.

	<b>Event کلید منو را ارسال می کند.</b>
clickOnText(text)	در رابط کاربری اپلیکیشن جاری، مقدار متّنی که به عنوان پارامتر به تابع ارسال شده را در خروجی برمی گرداند.
enterText()	یک متّن ساده در رابط کاربری جاری وارد می کند.
searchText(text)	در رابط کاربری جاری، متّنی که به عنوان پارامتر به تابع حاضر ارسال شده را جستجو کرده و در صورت یافتن، مقدار بولی true را بازمی گرداند.
searchButton(text)	کنترل دکمه ای که مقدار متّن (text) آن با مقدار پارامتر ارسالی منطبق می باشد را در رابط کاربری جاری پیدا می کند.
clickOnSearch()	امکان کلیک بر روی بخشی از نمایشگر را به شما می دهد.
goBack()	دکمه هی بازگشت را فشار می دهد.
setDatePicker()	یک تاریخ در کنترل DatePicker اندروید انتخاب می کند.
clickInList(x); pressSpinnerItem(0,2);	بر روی آیتم در لیست ساده ListView کلیک می کند. یک آیتم در لیست کشویی Spinner را فشار می دهد.
isCheckBoxChecked()	بررسی می کند آیا چک باکس تیک دار و فعال شده است یا خیر.
takeScreenshot()	یک عکس (screenshot) از وضعیت جاری تهیه کرده و داخل حافظه هی دستگاه میزبان در آدرس /sdcard/Robotium-Screenshots/ لازم است جهت فراخوانی این تابع، مجوز theandroid.permission.WRITE_EXTERNAL_STORAGE را در فایل تنظیمات AndroidManifest.xml مربوط به اپلیکیشن مورد تست اعلان نمایید.
waitForActivity(SecondActivity.class, 2000)	به مدت 2 ثانیه منتظر activity ارسال شده به تابع به عنوان پارامتر می ماند (چنانچه در مدت معین شده یافت نشد، مقدار بولی false را بازمی گرداند).

می توانید با فراخوانی متد solo.setActivityOrientation(Solo.LANDSCAPE) وضعیت نمایش صفحه هی جاری (activity) (orientation) را تنظیم نمایید.

جهت تست رشته های ترجمه شده و بین المللی (internationalized strings)، می بایست با فراخوانی متد (getString(id) به فایل resources از پروژه مورد تست دسترسی پیدا کنید.

مثال:

```
// Here we use the ID of the string to find the right button
solo.clickOnButton(solo.getString(de.vogella.android.test.target.R.string.button1));
```

## اجرای تست

تست Robotium را به همان شیوه ای که محیط کاری Eclipse را راه اندازی می کنید، با کلیک راست بر روی test class (کلاس آزمایشی) و انتخاب آیتم Run-As ▶ Android JUnit Test از منو، اجرا نمایید.

در صورت تمایل می توانید تست های Robotium را با درج دستور زیر در پنجره فرمان، اجرا نمایید.

```
adb shell am instrument  
-w de.vogella.android.test.tester/android.test.InstrumentationTestRunner
```

## تمرین: تست نویسی با توابع Robotium

### تمرین: نوشتن یک تست واقعی با Robotium

یک پروژه ای جدید اندروید به نام com.vogella.android.test.robotium.target ایجاد نمایید. در صفحه ای اول سه کنترل دکمه قرار دهید. پس از کلیک بر روی دکمه، با توجه به کدی که نوشته اید، می بایست activity (صفحه) دیگری فراخوانی شده و یک لیست را برای کاربر به نمایش بگذارد.

## 7-43-1 ساخت پروژه‌ی آزمایشی و اضافه کردن کتابخانه‌ی Robotium به

آن

یک پروژه‌ی جدید جهت تست به نام com.vogella.android.test.robotium.targetTest ایجاد نمایید. سپس داخل آن پوشه‌ی جدیدی به نام libs ایجاد کرده و فایل Robotium JAR را در آن جایگذاری نمایید.

**نکته:** پس از جایگذاری فایل JAR در پوشه‌ی نام برد، مجموعه ابزار Android (android tooling) محیط کاری Eclipse خود به صورت اتوماتیک آن را به build path پروژه‌ی شما اضافه می‌کند. لازم به ذکر است که قرار دادن فایل JAR در پوشه‌ای با اسم متفاوت، غالباً سبب رخداد خطای ClassNotFoundException برای کلاس Solo می‌شود.

## 7-43-2 ایجاد پروژه‌ی آزمایشی و افزودن کتابخانه‌ی Robotium

کلاس آزمایشی (test class) زیر را پیاده سازی نمایید.

```
package de.vogella.android.test.target.test;
import junit.framework.Assert;
import android.test.ActivityInstrumentationTestCase2;
import com.robotium.solo.Solo;
import de.vogella.android.test.target.SimpleActivity;
import de.vogella.android.test.target.SimpleListActivity;
public class SimpleActivityTest extends
        ActivityInstrumentationTestCase2<SimpleActivity> {
    private Solo solo;
    public SimpleActivityTest() {
        super(SimpleActivity.class);
    }
    public void setUp() throws Exception {
        solo = new Solo(getInstrumentation(), getActivity());
    }
    @Override
    public void tearDown() throws Exception {
        solo.finishOpenedActivities();
    }
    public void testListItemClickShouldDisplayToast() throws Exception {
        // check that we have the right activity
    }
}
```

```

solo.assertCurrentActivity("wrong activity", SimpleActivity.class);
// Click a button which will start a new Activity
// Here we use the ID of the string to find the right button
solo.clickOnButton(solo
    .getString(de.vogella.android.test.target.R.string.button1));
// assert that the current activity is the SimpleListActivity.class
solo.assertCurrentActivity("wrong activity", SimpleListActivity.class);
solo.clickInList(1);
// searchForText has a timeout of 5 seconds
assertTrue(solo.waitForText("Android")); // Assertion
solo.clickInList(2);
assertTrue(solo.waitForText("iPhone")); // Assertion
solo.clickInList(3);
assertTrue(solo.waitForText("Blackberry")); // Assertion
solo.goBack();
solo.clickOnButton("Button2");
solo.clickOnButton("Button3");
}
public void testListItemClickShouldDisplayToast() throws Exception {
    // open the menu
    solo.sendKey(Solo.MENU);
    solo.clickOnText("Preferences");
    solo.clickOnText("User");
    solo.clearEditText(0);
    Assert.assertTrue(solo.searchText(""));
    solo.enterText(0, "http://www.vogella.com");
    Assert.assertTrue(solo.searchText("http://www.vogella.com"));
    solo.goBack();
}
}

```

## برطرف کردن خطاهای اپلیکیشن

فرض کنید تست با تنظیمات و پیاده سازی شما از عملکرد صحیح برخوردار می باشد ولی اپلیکیشن نتیجه‌ی مورد انتظار را ارائه نمی دهد. اپلیکیشن را طوری ویرایش کنید که تست با موفقیت انجام شود.



بخش نهم :

## استفاده از فریم ورک 2 Dagger جهت تزریق نیازمندی ها به پروژه (کلاس ها و کتابخانه های مورد نیاز)/ **Dagger 2 & Dependency injection**

مقاله‌ی حاضر به شرح نحوه‌ی استفاده از فریم ورک 2 Dagger جهت تزریق نیازمندی‌ها به پروژه به آن در اپلیکیشن‌های متعارف جاوا و اندروید می‌پردازد.

## 44-مقدمه ای بر مفهوم dependency injection استفاده از dependency injection در محیط و بستر اجرای Java

### Dependency injection-7-44-1 چیست؟

امروزه مفهومی است که در بسیاری از زبان های برنامه نویسی پیاده سازی شود. مفهوم کلی تری که dependency injection بر آن پایه ریزی شده است، Inversion of Control می باشد. در مهندسی نرم افزار، وارونگی کنترل گونه ای از توسعه و طراحی اپلیکیشن را توصیف می کند که بخش های اختصاصی نوشته شده یک برنامه رایانه ای جریان کنترل را از یک کتابخانه با قابلیت باز استفاده عمومی دریافت می کند. معماری نرم افزاری ای که از این طرح به وجود می آید در مقایسه با برنامه نویسی سنتی رویه ای کنترل را وارونه می سازد: در برنامه نویسی سنتی کد اختصاصی که هدف برنامه را بیان می کند، برای مراقبت از وظایف عمومی کتابخانه های باز قابل استفاده را فرا می خواند اما با ورآونه سازی کنترل این کدهای دارای قابلیت باز استفاده هستند که کدهای اختصاصی یا کدهای مربوط به وظیفه خاص را صدا می زند. وارونگی کنترل برای افزایش ویژگی پودمانی / بیمانه ای (modularity) برنامه و قابلیت گسترش آن استفاده شده و در پارادایم برنامه نویسی شی گرا و دیگر الگوهای برنامه نویسی هم مورد استفاده قرار می گیرد.

Dependency Injection (فراهم نمودن نیازمندی های یک آبجکت) در برنامه نویسی شی گرا، الگوی توسعه است با قاعده ای اصلی جدا کردن رفتار از تحلیل نیازمندی (Dependency = روشه برای تجزیه کردن اجزا کاملا مستقل نرم افزاری. به صورت خلاصه Dependency injection، الگویی است جهت تزریق نیازمندی ها و آبجکت های مورد نیاز خارجی یک کلاس به آن، بجای استفاده مستقیم از آن نیازمندی ها (فراهم کردن آبجکت مورد نظر) در داخل کلاس. با توجه به این مفهوم، یک کلاس نباید نیازمندی ها (dependency) خود را به صورت تنظیم کند، بلکه می بایست آن را از بیرون و توسط کلاس های دیگر تامین نماید.

چنانچه یک کلاس جاوا از نمونه‌ی کلاس دیگر استفاده کند، گفته می‌شود که به آن کلاس وابستگی دارد. در اصطلاح این امر class dependency (وابستگی یک کلاس به کلاس خارجی) خوانده می‌شود. به عنوان مثال، کلاسی که به قابلیت‌های سرویس logger (ثبت گزارش) احتیاج داشته باشد، در اصطلاح گفته می‌شود که کلاس اول به کلاس سرویس دهی (service) احتیاج دارد یا کلاس اول قابلیت‌ها و نیازمندی‌های خود را از کلاس خارجی (class dependency) تأمین می‌کند.

در حالت ایده‌آل، کلاس‌های Java بایستی تا حد امکان از دیگر کلاس‌های Java مستقل باشند. از این طریق، امکان استفاده‌ی مجدد از کلاس‌ها افزایش یافته و متعاقباً می‌توان این کلاس‌ها را به صورت ایزووله یا مجزا از یکدیگر به راحتی تست نموده و از عملکرد صحیح آن‌ها اطمینان حاصل کرد.

چنانچه کلاس Java به واسطه‌ی کلیدوازه‌ی new، نمونه‌ای از کلاس دیگر را ایجاد کرده و مورد استفاده قرار دهد، در آن صورت نمونه‌ی ایجاد شده از کلاس قابل استفاده و تست به طور مجزا از کلاس میزبان نخواهد بود. از این رخداد تحت عنوان hard dependency (وابستگی شدید) یاد می‌شود.

```
package com.example.e4.rcp.todo.parts;
import java.util.logging.Logger;
public class MyClass {
    private final static Logger logger;
    public MyClass(Logger logger) {
        this.logger = logger;
        // write an info log message
        logger.info("This is a log message.")
    }
}
```

**نکته:** لازم به توضیح است که کلاس حاضر، صرفاً یک کلاس ساده‌ی جاوا بوده (هیچ ویژگی خاصی ندارد)، جز اینکه از ایجاد مسقیم آبجکت جلوگیری می‌کند.

کلاس framework که گاه dependency container نیز خوانده می‌شود، قادر است نیازمندی‌های این کلاس را به راحتی تحلیل و فراهم نماید (Dependency injection container) یک کلاس است که می‌داند چگونه آبجکت‌ها و متعاقباً آبجکت‌های وابسته به آن را نمونه‌سازی و تنظیم نماید. با

این تجزیه و تحلیل، کلاس قادر خواهد بود یک نمونه از کلاس مورد نظر ایجاد کرده و سپس آبجکت های مورد نیاز را به واسطهٔ `java reflection`، به داخل `dependency` های مشخص شده تزریق کند.

با این روش کلاس جاوا دیگر هیچ وابستگی شدید (`hard dependency`) ندارد، بدین معنی که دیگر برای تامین قابلیت ها و نیازمندی های خود به نمونه ای از کلاس دیگر احتیاج ندارد. این امکان به توسعه دهنده اجازه می دهد تا کلاس خود را، برای مثال با استفاده از آبجکت های ساختگی (`mock`)، به صورت ایزوله آزمایش نماید.

آبجکت های ساختگی (`mock`) آبجکت هایی هستند که قابلیت و رفتار آبجکت واقعی را شبیه سازی می کنند و در واقع سعی دارند تا مانند آبجکت مورد نظر رفتار کنند. اما این آبجکت های ساختگی یا `mock` برنامه ریزی نمی شوند بلکه طوری تنظیم شود که همیشه به یک شیوه ای از پیش تعیین شده و مشخص رفتار کنند. واژهٔ `mock` یک کلمهٔ انگلیسی است که معنی شبیه سازی و تقلید را در فارسی می دهد. بنابراین یک آبجکت `mock`، همان طور که قبل از ذکر شد، رفتار یک آبجکت واقعی را شبیه سازی و تقلید می کند.

در صورت پیاده سازی الگو توسعهٔ DI در پروژه، کلاس های جاوا به راحتی به صورت ایزوله قابل تست خواهد بود.

## ۴۴-۷- استفاده از annotation ها جهت توصیف و شرح نیازمندی ها / dependency های کلاس

روش های مختلفی برای شرح و مشخص کردن `dependency` های یک کلاس وجود دارد. پرکاربردترین روش معمولاً در توصیف مسقیم `dependency` ها در کلاس میزبان با استفاده از `Java` خلاصه می شود (نشانه و علامت گذاری نیازمندها داخل کلاس با استفاده از annotation ها).

annotation های متعارف جاوا که برای شرح و نشانه گذاری های یک کلاس مورد استفاده قرار می گیرند، همگی در JSR330 اعلام شده اند. JSR330 دو دستور `@Inject` و `@Named` را برای نشانه گذاری نیازمندی ها در کلاس، در اختیار توسعه دهنده قرار می دهد.

کد زیر یک کلاس را نمایش می دهد که با استفاده از annotation ها، نیازمندی ها و dependency های خود را مستقیماً داخل کلاس میزبان اعلام می نماید.

```
// import statements left out
public class MyPart {
    @Inject private Logger logger;
    // inject class for database access
    @Inject private DatabaseAccessClass dao;
    @Inject
    public void createControls(Composite parent) {
        logger.info("UI will start to build");
        Label label = new Label(parent, SWT.NONE);
        label.setText("Eclipse 4");
        Text text = new Text(parent, SWT.NONE);
        text.setText(dao.getNumber());
    }
}
```

**توجه:** همان طور که در کد بالا مشاهده می کنید، کلاس حاضر از کلیدواژه `new` برای نمونه سازی و ساخت کامپوننت های رابط کاربری (UI) استفاده می کند. با این کار برنامه نویس اعلان می دارد که کامپوننت های رابط کاربری (اجزایی که با عملگر `new` نمونه سازی شده اند) قرار نیست در آینده تست شوند یا با آبجکت های ساختگی و شبیه ساز تست ها جایگزین گردند. به عبارت دیگر با پیاده سازی فوق، برنامه نویس تصمیم گرفته که به toolkit و مجموعه ابزار UI مربوطه وابستگی شدید (hard coupling) داشته باشد.

### 3-44-7- بر اساس JSR330، آبجکت ها به کدام بخش از کلاس تزریق یا پاس داده می شوند؟

الگوی DI را می توان از روش های زیر در پروژه پیاده سازی کرد:

- از طریق تابع سازنده `i` (Constructor) کلاس که در اصطلاح `construction injection` (Constructor Injection) شناخته می شود.

- از طریق فیلد که در اصطلاح **field injection** نام دارد.
- از طریق پارامترهای یک متده که از آن تحت عنوان **method injection** یاد می شود.

می توان از **dependency injection** برای فیلدها و متدهای **static** و غیر **static** نیز استفاده کرد. لازم به ذکر است که توسعه دهنگان خبره سعی می کنند از پیاده سازی DI بر روی متدها و فیلدهای **static** خودداری کنند چرا که علاوه بر به همراه داشتن محدودیت هایی، اشکال زدایی و فرایند **debug** را بسیار سخت می کند.

- فیلدهای **static** پس از اینکه اولین آبجکت کلاس از طریق پیاده سازی DI ساخته شد، تزریق می شوند، بدین معنی که داخل تابع سازنده (**constructor**) دسترسی به آن ها امکان پذیر نخواهد بود.
- فیلدهای **static** را نمی توان به عنوان **final** علامت گذاری کرد چرا که در این صورت، کامپایلر یا خود اپلیکیشن در زمان اجرای برنامه ایراد می گیرد.
- متدهای **static** تنها یکبار پس از اینکه اولین نمونه از کلاس ایجاد شد، فراخوانی می شوند.

#### 4-7-ترتیبی که DI بر اساس آن در کلاس اجرا و پیاده سازی می شود

بر اساس JSR330 الگوی DI به ترتیب زیر در کلاس پیاده سازی می شود:

- تزریق نیازمندی ها از طریق تابع سازنده کلاس (Constructor injection).
- تزریق از طریق فیلد کلاس (field injection).
- تزریق با ارسال پارامتر به متده کلاس (method injection).

JL، ترتیبی که متدها یا فیلدهای نشانه گذاری شده با `@Inject` بر آن اساس اجرا می شوند را تعیین نمی کند. به عبارت دیگر شما نمی توانید فرض را بر این بگذارید که متدها یا فیلدها به همان ترتیبی که در کلاس اعلام شده اند، فراخوانی می شوند.

**توجه:** از آنجایی که فیلدها و پارامترهای متدهای پس از فراخوانی تابع سازنده (constructor) به داخل کلاس تزریق می شوند، امکان استفاده از متغیرهای عضو تزریق شده از بیرون در تابع سازنده وجود ندارد.

## 45-7 فریم ورک های جاوا و dependency injection

می توانید با تعییه constructor ها و توابع getter/setter کافی داخل کلاس های پروژه خود، بدون استفاده از framework، الگو توسعه دی را پیاده سازی نمایید.

یک فریم ورک که برای پیاده سازی الگو DI طراحی شده است، صرفا مقداردهی اولیه کلاس ها با آبجکت های مناسب را ساده تر می سازد.

دو فریم ورک پرطرفدار جهت پیاده سازی DI عبارتند از Spring و Google Guice

## 46-7 Dagger2 با فریم ورک Dependency injection

### Dagger2-7-46-1 چیست؟

همان طور که قبل گفته شد، به این خاطر که ایزووله سازی آبجکت ها و فراهم کردن نیازمندی هایشان (dependency) به سادگی صورت پذیرد، میتوان از فریمورک های مختلفی استفاده کرد که قطعاً یکی از پرطرفدارترین و کارترین آن ها است. Dagger یک فریم ورک جهت پیاده

سازی الگو توسعه‌ی DI در پروژه محسوب می‌شود که مبتنی بر Java Specification Request (JSR 330) طراحی شده، از ابزار code generation استفاده می‌کند و مبتنی بر annotation ها می‌باشد. کد تولید شده خوانا بوده و اشکال زدایی آن بسیار آسان می‌باشد.

از دو annotation زیر استفاده می‌کند:

- @Provides و @Module: کلاس و متدهایی تعریف می‌کند که نیازمندی‌های کلاس‌ها را در اختیارشان قرار می‌دهد.
- @Inject dependency: هایی را فراخوانی یا درخواست می‌کند. می‌توان از این دستور برای نشانه گذاری Constructor، یک فیلد یا متد از کلاس استفاده کرد.
- @Component: ماثول‌های انتخابی را فعال ساخته و برای پیاده‌سازی DI در پروژه استفاده می‌شود.

در فریم‌ورک Dagger اجازه‌ی استفاده از فیلدهایی که به صورت private تعریف شده‌اند برای وجود ندارد چرا که generated code 2 از Dagger (reflection field injection) برای دسترسی به فیلدها استفاده می‌کند.

## 7-46-2-اعلان dependency provider ها (تعریف ارائه‌دهندگان آبجکت های موردنیاز و نیازمندی‌های کلاس)

منظور از واژه‌ی dependency injection context، تعداد آبجکت‌هایی است که قابلیت تزریق و ارسال آن‌ها به عنوان پارامتر به کلاس‌ها وجود دارد.

در فریم‌ورک Dagger2، کلاس‌هایی که با دستور @Module نشانه گذاری شده‌اند، وظیفه‌ی ارائه‌ی آبجکت‌هایی را دارند که قرار است به عنوان dependency و پارامتر در اختیار کلاس

درخواست کننده قرار گیرد. این کلاس های میزبان می توانند در خود توابعی داشته باشند که با دستور `@Provides` نشانه گذاری شده و در خروجی آبجکت هایی را برمی گردانند که می توان از آن ها برای رفع نیازمندی های کلاس بهره گرفت.

متدهایی که با دستور `@Provides` نشانه گذاری شده باشند، این قابلیت را دارند که کلاس های مورد نیاز یا dependency ها را از طریق پارامترهای ارسالی به متدهای فراهم کنند. این آبجکت های مورد نیاز یا dependency ها را با فریم ورک Dagger2 در اختیار کلاس درخواست کننده قرار می دهیم.

### 46-3-7- تعریف آبجکت های مورد نیاز یا dependency ها (Object)

جهت اعلان یک dependency کافی است ابتدا دستور `@Inject` را بالای کد مربوطه درج نمایید. پس از اینکه تابع سازنده (constructor) را با دستور مذکور نشانه گذاری کردید، Dagger2 قادر خواهد بود که با استفاده از نمونه‌ی این آبجکت (instance) نیازمندی‌ها یا به اصطلاح آبجکت های مورد نیاز کلاس را برآورده سازد. علت استفاده از روش مذکور برای پیاده‌سازی DI این است که از نوشتمن چندین متدهای نشانه گذاری شده با `@Provides` و طولانی سازی کد خودداری شود.

### 46-4-7- برقراری ارتباط بین Consumer ها (کلاس های درخواست کننده یا مصرف کننده) و provider های (ارائه دهنده) آبجکت های مورد نیاز

دستور `@Component` در سطح یک interface (الگوی پیاده‌سازی) مورد استفاده قرار می گیرد. فریم ورک Dagger2 با استفاده از این interface، کد لازم را تولید می نماید. ساختار نحوی برای پیاده‌سازی کلاس مورد نیاز (generated class) این است که Dagger به عنوان پیشوند درج شده و بلافاصله پس از آن اسم interface مربوطه قرار می گیرد. کلاسی که در نتیجه‌ی استفاده از الگوی فوق تولید می شود یک متدهای `create` در اختیار توسعه دهده قرار می دهد که به واسطه‌ی

آن می توان آبجکت ها را بر اساس configuration ارائه شده، تنظیم کرد. متدهایی که در سطح اعلان شده اند می توانند به این آبجکت های تعریف شده دسترسی داشته باشند.

اینترفیس @Component ارتباط بین ارائه دهنده آبجکت ها و نیازمندی های کلاس (module) ها) و آبجکت هایی که نیاز به dependency را اعلان می کنند، برقرار می نماید. جدول زیر توضیح مختصر و مفیدی پیرامون annotation های Dagger ارائه می نماید.

دستور/annotation	شرح کاربرد
@Module	در سطح کلاس هایی تعریف می شود که حامل متدهای نشانه گذاری شده با دستور @Provides می باشند.
@Provides	می تواند بر روی متدهای موجود در کلاس نشانه گذاری شده با دستور @Module مورد استفاده قرار گیرد. همچنین برای متدهایی کاربرد دارد که آبجکت های موردنیاز کلاس را برای پیاده سازی الگو توسعه دی در اختیار آن قرار می دهد.
@Singleton	با درج این دستور یک نمونه ی واحد از آبجکت ارائه شده، ساخته و به اشتراک گذاشته می شود.
@Component	در سطح یک interface مورد استفاده قرار می گیرد. این interface را فریم ورک Dagger2 برای تولید کدی که با استفاده از ماژول، نیازمندی ها و آبجکت های مورد درخواست کلاس را فراهم می کند، مورد استفاده قرار می گیرد.

## Scope annotation-7-46-5

می توانید با استفاده از دستور @Singleton اعلان نمایید که از آبجکت مورد نظر یک نمونه بیشتر ایجاد نخواهد شد.

## 7-46-6-برخورد ویژه دی (field injection) با فیلد ها

Dagger 2 خود به صورت اتوماتیک فیلد ها را به داخل کلاس تزریق نمی کند. همچنین از تزریق فیلد هایی که با سطح دسترسی private تعریف شده اند، عاجز است. اگر می خواهید از روش

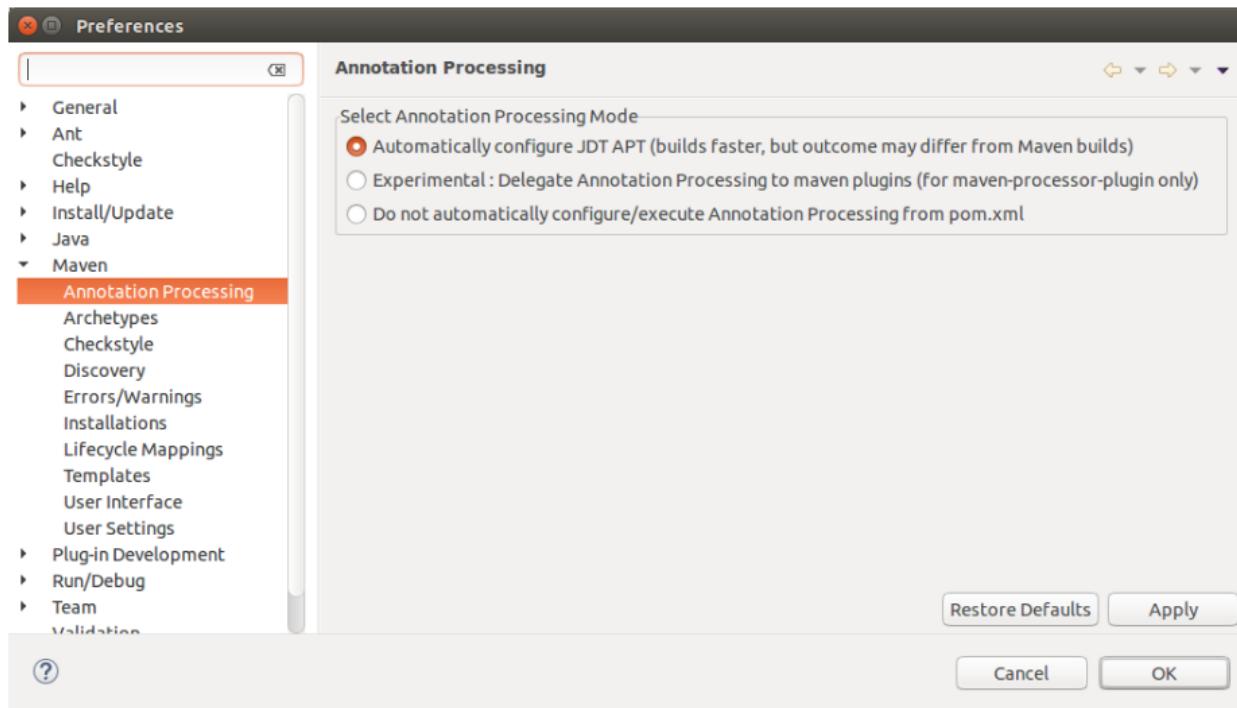
برای برطرف کردن نیازمندی استفاده نمایید، در آن صورت می بایست یک متدهای اعلان نموده که نمونه ای که قرار است فیلد را به داخل آن تزریق کنید به عنوان پارامتر بپذیرد.

## 7-47-7-استفاده از Dagger در محیط کاری Eclipse همراه با سیستم

### Maven کامپایل

جهت استفاده از محیط کاری Eclipse و سیستم کامپایل Maven با فریم ورک DI به نام `maven-eclipse` لازم است ابزار Maven و افزونه `apt` را در محیط برنامه نویسی مزبور نصب کنید تا بدین وسیله Maven قابلیت تنظیم و آماده سازی پردازش گر `annotation processor` ها را داشته باشد. برای فراهم آوردن قابلیت پشتیبانی Eclipse از سیستم Maven، بایستی به سایت آپدیت و بروز رسانی ویرایش مربوطه از Eclipse مراجعه نموده و پس از بارگیری از آدرس <http://download.jboss.org/jbosstools/updates/m2e-extensions/m2e-apt> آن را در محیط کاری مورد نظر نصب نمایید.

پس از نصب افزونه `apt`، لازم است با طی کردن این مسیر داخل preferences (تنظیمات `Window > Preferences > Maven > Annotation Processing`)، پردازش مناسب (قابلیت پردازش `annotation processing`) را فعال سازی نمایید. حال گزینه `Automatically configure JDT` را انتخاب کنید.



## تمرین کاربردی: پیاده سازی الگو توسعه DI از طریق فریم ورک Dagger2

در تمرین حاضر شما می باشید از فریم ورک Dagger 2 در یک برنامه می متعارف جاوا که با سیستم کامپایل مدیریت Maven می شود، جهت پیاده سازی الگو توسعه DI استفاده نمایید.

این پروژه می آزمایشی را می توانید در هر کدام از محیط های برنامه نویسی Java که مایلید پیاده سازی کنید. در تمرین جاری ما از محیط کاری Eclipse که ابزار Maven بر روی آن نصب شده، استفاده می کنیم. همچنین در صورتی که IDE نام بردۀ را در اختیار ندارید، می توانید پروژه را از طریق یک text editor معمولی و با فراخوانی دستورات Maven در پنجره فرمان (command line) توسعه داده و کامپایل نمایید.

چنانچه قصد استفاده از محیط کاری Eclipse را دارید، در آن صورت لازم است ابزار لازم Maven را نیز دانلود و نصب نمایید.

## 7-46-7-ساخت پروژه‌ی آزمایشی و استفاده از فریم‌ورک 2 Dagger

یک پروژه‌ی جدید Maven با اسم پکیج com.vogella.java.dagger2 ایجاد نمایید.

## 7-46-8-تعریف یا تنظیم و ویرایش فایل Maven build

محتوای فایل pom.xml را به صورت زیر ویرایش نمایید.

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.vogella.java.dagger2</groupId>
  <artifactId>com.vogella.java.dagger2</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>com.vogella.java.dagger2</name>
  <dependencies>
    <dependency>
      <groupId>com.google.dagger</groupId>
      <artifactId>dagger</artifactId>
      <version>2.4</version>
    </dependency>
    <dependency>
      <groupId>com.google.dagger</groupId>
      <artifactId>dagger-compiler</artifactId>
      <version>2.4</version>
      <optional>true</optional>
    </dependency>
  </dependencies>
  <build>
    <plugins>
      <plugin>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.3</version>
        <configuration>
          <source>1.8</source>
          <target>1.8</target>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```

## 7-46-9-تعریف و استفاده از کلاس هایی که به آبجکت هایی از بیرون نیاز دارند (درخواست dependency دارند)

دو کلاس زیر را تعریف کنید که برای عملکرد خود نیاز به آبجکت هایی از بیرون دارند (dependency) هایی را از بیرون درخواست یا فراخوانی می کنند.

```
package com.vogella.java.dagger2;
import javax.inject.Inject;
public class User {
    private String firstName;
    private String lastName;
    public User(String firstName, String lastName) {
        this.firstName = firstName;
        this.lastName = lastName;
    }
    @Override
    public String toString() {
        return "User [firstName=" + firstName + ", lastName=" + lastName + "]";
    }
}
package com.vogella.java.dagger2;
import javax.inject.Inject;
public class BackendService {
    @Inject
    public User user;
    private String serverUrl;
    @Inject
    public BackendService(@Named("serverUrl") String serverUrl) {
        this.serverUrl = serverUrl;
    }
    public boolean callServer() {
        if (user != null && serverUrl != null && serverUrl.length() > 0) {
            System.out.println("User: " + user + " ServerUrl: " + serverUrl);
            return true;
        }
        return false;
    }
}
```

## 7-46-10-تعریف کامپوننت های مورد نیاز (دستور @Component)

این کامپوننت ها هستند که اعلان می کنند کدام module یا کامپوننت های نرم افزاری هستند که dependency ها و آبجکت های مورد نیاز را فراهم می کنند. Dagger 2 با استفاده از این accessor را ایجاد می کند. این کلاس نیز متدهای آماده و تعریف شده در بدنه interface را برای پیاده سازی ارائه می دهد.

```

package com.vogella.java.dagger2.component;
import javax.inject.Singleton;
import com.vogella.java.dagger2.BackendService;
import com.vogella.java.dagger2.modules.BackEndServiceModule;
import com.vogella.java.dagger2.modules.UserModule;
import dagger.Component;
@Singleton
@Component(modules = { UserModule.class, BackEndServiceModule.class })
public interface MyComponent {
    // provide the dependency for dependent components
    // (not needed for single-component setups)
    BackendService provideBackendService();
    // allow to inject into our Main class
    // method name not important
    void inject(Main main);
}

```

متدی که امکان تزریق و مقداردهی فیلدها در BackendServer را می دهد.

## 7-46-11-Dagger - استفاده از کد تولید شده توسط

کلاس آزمایشی زیر به صورت کاربردی از کدی که فریم ورک Dagger به صورت آماده در اختیار آن قرار می دهد استفاده می نماید.

```

package com.vogella.java.dagger2.main;
import com.vogella.java.dagger2.BackendService;
import com.vogella.java.dagger2.component.DaggerMyComponent;
import com.vogella.java.dagger2.component.MyComponent;
public class Main {
    @Inject
    BackendService backendService; //
    private MyComponent component;
    private Main() {
        component = DaggerMyComponent.builder().build();
        component.inject(this);
    }
    private void callServer() {
        boolean callServer = backendService.callServer();
        if (callServer) {
            System.out.println("Server call was successful. ");
        } else {
            System.out.println("Server call failed. ");
        }
    }
    public static void main(String[] args) {
        Main main = new Main();
        main.callServer();
    }
}

```

```

    }
}

```

از آنجایی که فیلدها به صورت اتوماتیک تزریق نمی شوند، متدهای مربوطه فراخونی شده و فیلدهای مورد نظر را مقداردهی می نماید.

## 47-7 داده های DI در پروژه های اندروید و Dagger 2

### 7-47-1 پیاده سازی الگو توسعه DI در پروژه های اندرویدی

بسیاری از کامپوننت های نرم افزاری اندروید همچون activity ها را خود فریم ورک اندروید نمونه سازی می کند. به عبارتی دیگر کار توسط توسعه دهنده به صورت دستی انجام نمی گیرد. این امر فراهم کردن آبجکت های مورد نیاز کلاس (dependency) و کامپوننت های نرم افزاری اندروید از بیرون به وسیله ای توابع سازنده (constructor) را پیچیده و دشوار می کند.

### 7-47-2 استفاده ای کاربردی از Dagger 2 در اندروید

جهت فعال سازی Dagger 2 و امکان استفاده از آن در پروژه، لازم است فایل build.gradle را ویرایش نمایید.

```

buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:2.2.0-alpha3'
        classpath 'com.neenbedankt.gradle.plugins:android-apt:1.8'
    }
}
allprojects {
    repositories {
        jcenter()
    }
}

```

```

        }
    }
task clean(type: Delete) {
    delete rootProject.buildDir
}

```

المان های زیر را به فایل app/build.gradle اضافه نمایید.

```

apply plugin: 'com.android.application'
apply plugin: 'com.neenbedankt.android-apt'
android {
    compileSdkVersion 22
    buildToolsVersion "23.0.0"
    defaultConfig {
        applicationId "com.vogella.android.dagger2simple"
        minSdkVersion 22
        targetSdkVersion 22
        versionCode 1
        versionName "1.0"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}
ext {
    JUNIT_VERSION = '4.12'
    DAGGER_VERSION ='2.4'
}
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.google.dagger:dagger:2.4'
    apt "com.google.dagger:dagger-compiler:DAGGER_VERSION"
    provided 'javax.annotation:jsr250-api:1.0'
    compile 'javax.inject:javax.inject:1'
    testCompile "junit:junit:$JUNIT_VERSION"
    testApt "com.google.dagger:dagger-compiler:DAGGER_VERSION"
}

```

**نکته:** لازم به ذکر است که بایستی apt را برای هر scope ای که Dagger است در آن مورد استفاده قرار بگیرد، دقیق تنظیم و مشخص نمایید. به طور مثال، برای تست اپلیکیشن های معمولی apt، برای تست هایی که بر روی دستگاه مجازی جاوا/JVM اجرا می شود testApt و در نهایت برای تست های اندرویدی نیز از testAndroidApt استفاده نمایید.

## تمرین کاربردی: پیاده سازی الگو توسعه‌ی DI با استفاده از فریم ورک در پروژه‌های اندرویدی Dagger 2

### هدف از تمرین

در این تمرین، ۲ دستورات عملی در یک پروژه‌ی اندرویدی جهت پیاده سازی DI مورد استفاده قرار می‌گیرد. در برنامه‌ی ساده‌ی بخش حاضر، یک کلاس activity تعریف می‌شود که قابلیت احراز هویت (authenticate) و بررسی صحت گذرواژه/نام کاربری (credentials) را به کاربر می‌دهد. مقدار خروجی و نتیجه‌ی پیاده سازی در کادر متن (text field) به نمایش گذاشته می‌شود.

### ساخت پروژه

یک پروژه‌ی جدید با اسم پکیج (com.vogella.android.dagger2simple) (top level package) ایجاد نمایید.

محتوای فایل layout ای که ظاهر کامپوننت activity و صفحه‌ی قابل مشاهده برای کاربر را مشخص می‌کند، به صورت زیر ویرایش نمایید.

```
<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"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/target"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />
</RelativeLayout>
```

## ۷-۴۷-۳- افزودن dependency ها و کتابخانه های مورد نیاز سیستم

### کامپایل Gradle

آیتم های زیر را به فایل build.gradle پردازه اضافه نمایید.

```
buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:2.2.0-alpha3'
        classpath 'com.neenbedankt.gradle.plugins:android-apt:1.8'
    }
}
allprojects {
    repositories {
        jcenter()
    }
}
task clean(type: Delete) {
    delete rootProject.buildDir
}
```

حال المان های زیر را به فایل app/build.gradle اضافه نمایید.

```
apply plugin: 'com.android.application'
apply plugin: 'com.neenbedankt.android-apt'
android {
    compileSdkVersion 22
    buildToolsVersion "23.0.0"
    defaultConfig {
        applicationId "com.vogella.android.dagger2simple"
        minSdkVersion 22
        targetSdkVersion 22
        versionCode 1
        versionName "1.0"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}
ext {
    JUNIT_VERSION = '4.12'
    DAGGER_VERSION ='2.4'
}
dependencies {
```

```

compile fileTree(dir: 'libs', include: ['*.jar'])
compile 'com.google.dagger:dagger:2.4'
apt "com.google.dagger:dagger-compiler:DAGGER_VERSION"
provided 'javax.annotation:jsr250-api:1.0'
compile 'javax.inject:javax.inject:1'
testCompile "junit:junit:$JUNIT_VERSION"
testApt "com.google.dagger:dagger-compiler:DAGGER_VERSION"
}

```

## 7-47- ترسیم یا اعلان نمودار dependency ها و نمایش وابستگی آبجکت

های مورد نیاز به یکدیگر

کلاس های زیر را جهت ارائه کردن dependency ها و متصل کردن آن ها به یکدیگر (آبجکت های مورد نیاز جهت رفع نیازمندی ها) در پروژه ایجاد نمایید.

```

package com.vogella.android.dagger2simple;
public class NetworkApi {
    public boolean validateUser(String username, String password) {
        // imagine an actual network call here
        // for demo purpose return false in "real" life
        if (username == null || username.length() == 0) {
            return false;
        } else {
            return true;
        }
    }
}

package com.vogella.android.dagger2simple.modules;
import com.vogella.android.dagger2simple.NetworkApi;
import javax.inject.Singleton;
import dagger.Module;
import dagger.Provides;
@Module
public class NetworkApiModule {
    @Provides
    @Singleton
    public NetworkApi getNetwork(){
        return new NetworkApi();
    }
}

package com.vogella.android.dagger2simple.components;
import android.app.Activity;
import com.vogella.android.dagger2simple.MainActivity;

```

```

import com.vogella.android.dagger2simple.NetworkApi;
import com.vogella.android.dagger2simple.modules.NetworkApiModule;
import javax.inject.Singleton;
import dagger.Component;
@Singleton
@Component(modules = {NetworkApiModule.class})
public interface DiComponent {
    // to update the fields in your activities
    void inject(MainActivity activity);
}

```

## 7-47-5 ساخت آبجکت پروژه (application)

در بدنه ی کلاس Application پروژه ی خود، کد زیر را پیاده سازی کنید. این کلاس بستر (context) بیاده سازی DI را فراهم کرده و سپس با فراخوانی تابع getComponent دسترسی به آن را مهیا می کند.

```

package com.vogella.android.dagger2simple;
import android.app.Application;
import com.vogella.android.dagger2simple.components.DaggerDiComponent;
import com.vogella.android.dagger2simple.components.DiComponent;
public class MyApplication extends Application {
    DiComponent component;
    @Override
    public void onCreate() {
        super.onCreate();
        component = DaggerDiComponent.builder().build();
    }
    public DiComponent getComponent() {
        return component;
    }
}

```

البته لازم است این کلاس را در لایه ی XML (داخل فایل تنظیمات اپلیکیشن Manifest) نیز ثبت و معرفی نمایید.

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.vogella.android.dagger2simple" >
    <application
        android:name="MyApplication"
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >

```

```

<activity
    android:name=".MainActivity"
    android:label="@string/app_name" >
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
</application>
</manifest>

```

فعالی مقادیر تولید شده را از طریق متدهی که در سطح کلاس `@Component` activity تعریف شده، تزریق می‌کند.

```

package com.vogella.android.dagger2simple;
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.TextView;
import android.widget.Toast;
import javax.inject.Inject;
public class MainActivity extends Activity {
    @Inject NetworkApi networkApi;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ((MyApplication) getApplication()).getComponent().inject(this);
        boolean injected = networkApi == null ? false : true;
        TextView userAvailable = (TextView) findViewById(R.id.target);
        userAvailable.setText("Dependency injection worked: " + String.valueOf(injected));
    }
}

```

## ۷-۴۷- تست اپلیکیشن و کسب اطمینان از عملکرد صحیح اپلیکیشن

اپلیکیشن را اجرا نمایید. UI اپلیکیشن می‌بایست صریحاً اعلان کند که با dependency injection موفقیت انجام شد. در غیر این صورت، نگاهی به کلاس‌های تولید شده در پوشه `apk` می‌باشد. این کلاس‌ها بایستی به طور خواناً سازماندهی شده و دقیقاً مشخص کنند که چه خطایی رخ داده است.



790

آدرس آموزشگاه : تهران - خیابان شریعتی - بالاتر از خیابان ملک - جنب بانک صادرات - پلاک 651 طبقه دوم - واحد 7

88146323 - 88446780 - 88146330

## 7-48-جايگزين کردن @Module با کلاس های اختصاصی خود در تست

می توانید در تست های نرم افزاری بجای کلاس های Dagger کلاس های اختصاصی خود را قرار دهید و این فریم ورک را طوری تنظیم کنید که از این کلاس ها استفاده کند. فریم ورک توابعی دارد که مازول مورد نظر را ساخته و مقدار دهی می کند. این امکان برای شما وجود دارد که با استفاده از builder فریم ورک مزبور کلاس های Dagger را با کلاس دلخواه خود (کلاسی که باید در تست مورد استفاده قرار گیرد) جایگزین نمایید.

این امکان در Mockito نیز قابل استفاده می باشد. فرض کنید می خواهید کلاس زیر را تست نمایید.

```
public class ImportantClass {
    private MyRestService restService;
    private MyLogger logger;
    @Inject public MainService(MyRestService restService, MyLogger logger) {
        this.restService = restService;
        this.logger = logger;
    }
    public void perform() {
        String s = restService.getData();
        logger.write(s.toUpperCase());
    }
}
```

می توانید این کلاس را با Mockito تست نمایید.

```
public class ImportantClassTest {
    public class MainServiceTest {
        @Rule public MockitoRule mockitoRule = MockitoJUnit.rule();
        @Mock MyRestService restService;
        @Mock MyLogger logger;
        @InjectMocks ImportantClass importClassToBeTested;
        @Test
        public void performShouldWriteOutput() {
            when(restService.getData()).thenReturn("abc");
            importClassToBeTested.perform();
            verify(logger).print("ABC");
        }
    }
}
```

## بخش دهم :

# ابزار تحلیل و سنجش کارایی اپلیکیشن های اندرویدی

آموزش حاضر به شرح ابزار موجود اندروید برای بررسی و سنجش کارایی اپلیکیشن های اندرویدی می پردازد.

## 49- مرور کلی

کارایی اپلیکیشن های اندرویدی در اجرای تمامی عملیات با سرعت هر چه تمام تر از اهمیت ویژه ای برخوردار است. مبحث جاری کلیه ای ابزار موجود اندروید در ره گیری و بهینه سازی اپلیکیشن را عنوان می کند.

## StrictMode-50-7

توسعه دهنده می بایست تا حد امکان از اجرای عملیات طولانی، سنگین و هزینه بر در thread اصلی (UI thread) خودداری کند. از جمله ای این عملیات می توان به دسترسی به فایل و اینترنت اشاره کرد.

برای دستیابی به این هدف می توان از StrictMode بهره گرفت. این حالت عملیاتی (StrictMode) از ورژن 9 توابع کتابخانه ای اندروید (API ورژن 9) برای اولین بار در چارچوب نرم افزاری اندروید معرفی شده و به توسعه دهنده این امکان را می دهد تا برای اپلیکیشن سیاست ها و قوانین خاصی حاکم بر مدیریت thread policy ها به نام thread policy تعریف کند.

به واسطه‌ی StrictMode توسعه دهد می‌تواند به سیستم اندروید دستور بدهد تا چنانچه اپلیکیشن عملیات سنگین و طولانی همچون دریافت ورودی/خروجی (I/O) را در thread اصلی به اجرا گذاشت، بلافاصله اپلیکیشن را متوقف سازد.

کد زیر نحوه‌ی استفاده از StrictMode را به نمایش می‌گذارد. به مجرد اینکه کامپوننت activity این تنظیمات و قوانین تنظیم شده را نقض کرد، سیستم اندروید آن را به طور ناگهانی متوقف می‌سازد.

```
package de.vogella.android.strictmode;
import java.io.BufferedWriter;
import java.io.OutputStreamWriter;
import android.app.Activity;
import android.os.Bundle;
import android.os.StrictMode;
public class TestStrictMode extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // Activate StrictMode
        StrictMode.setThreadPolicy(new StrictMode.ThreadPolicy.Builder()
            .detectAll()
            .detectDiskReads()
            .detectDiskWrites()
            .detectNetwork()
            // alternatively .detectAll() for all detectable problems
            .penaltyLog()
            .penaltyDeath()
            .build());
        StrictMode.setVmPolicy(new StrictMode.VmPolicy.Builder()
            .detectLeakedSqlLiteObjects()
            .detectLeakedClosableObjects()
            // alternatively .detectAll() for all detectable problems
            .penaltyLog()
            .penaltyDeath()
            .build());
        // Test code
        setContentView(R.layout.main);
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        String eol = System.getProperty("line.separator");
        try {
            BufferedWriter writer =
                new BufferedWriter(
                    new OutputStreamWriter(
                        openFileOutput("myfile",
                            MODE_WORLD_WRITEABLE)));

```

```
        writer.write("This is a test1." + eol);
        writer.write("This is a test2." + eol);
        writer.write("This is a test3." + eol);
        writer.write("This is a test4." + eol);
        writer.write("This is a test5." + eol);
        writer.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

رنگ StrictMode تنها در مرحله‌ی توسعه‌ی نرم افزار قابل استفاده می‌باشد. توصیه می‌شود از بکار بردن آن در یک اپلیکیشن توزیع و نصب شده که به صورت نهایی اجرا می‌شود، خودداری گردد.

بخش Developer Settings در Setting اپلیکیشن گوشی اندروید به شما این امکان را می دهد تا تنظیماتی را تعیین کنید که تحلیل و سنجش کارایی اپلیکیشن را آسان می سازد. به عنوان مثال شما می توانید اعلان کنید که محل برخورد انگشت با نمایشگر هایلایت شود.

برای مثال چنانچه ویرایش 4.2 اندروید بر روی دستگاه شما نصب است، برای دسترسی به بخش تنظیمات مربوط به توسعه دهنده، می بایست به بخش About در Settings مراجعه نموده و بر روی المان Build number هفت بار کلیک نمایید.

در صورتی که گوشی شما این گزینه‌ی تنظیم را نداشت، می‌توانید از برنامه‌ی شبیه‌ساز (emulator) استفاده نمایید.

در برخی موارد لازم است اپلیکیشن را مجددا راه اندازی کرده تا تنظیمات مورد نظر فعال شده و به درستی عمل کند.

مقدمة-7-50-1

Traceview عبارت است از یک graphical viewer که به شما امکان می دهد تا گزارشاتی (log ها) که اپلیکیشن اندروید ضبط می کند را مشاهده نمایید. به واسطهٔ TraceView شما می توانید کارایی اپلیکیشن را سنجیده و مشکلاتی که کارایی اپلیکیشن را پایین آورده و کیفیت تجربهٔ کاربری را پایین می آورد، شناسایی و پرطرف نمایید.

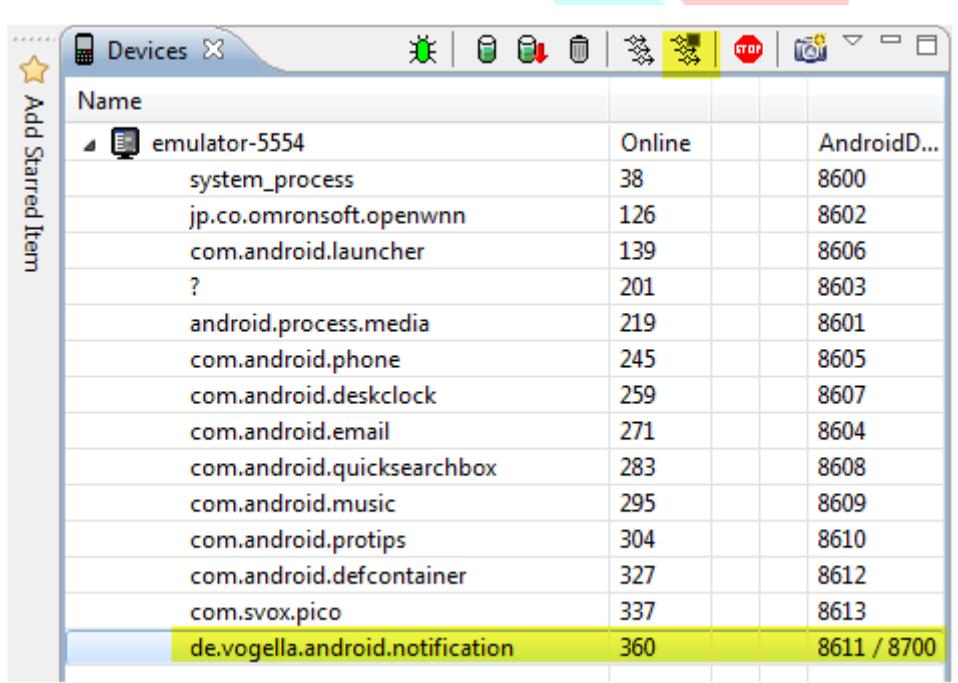
TraceView به عنوان یک ابزار مستقل در فolder tools از پوشه‌ی نصب ابزار توسعه و ساخت اپلیکیشن‌های اندرویدی / Android SDK installation جایگذاری شده و قابل دسترسی می‌باشد. توسعه دهنده می‌تواند با نصب ADT (مجموعه ابزار توسعه‌ی نرم افزار اندرویدی) آن را در محیط برنامه نویسی Eclipse جاسازی کند.

## 50-7- استفاده از ابزار TraceView در محیط کاری Android Studio

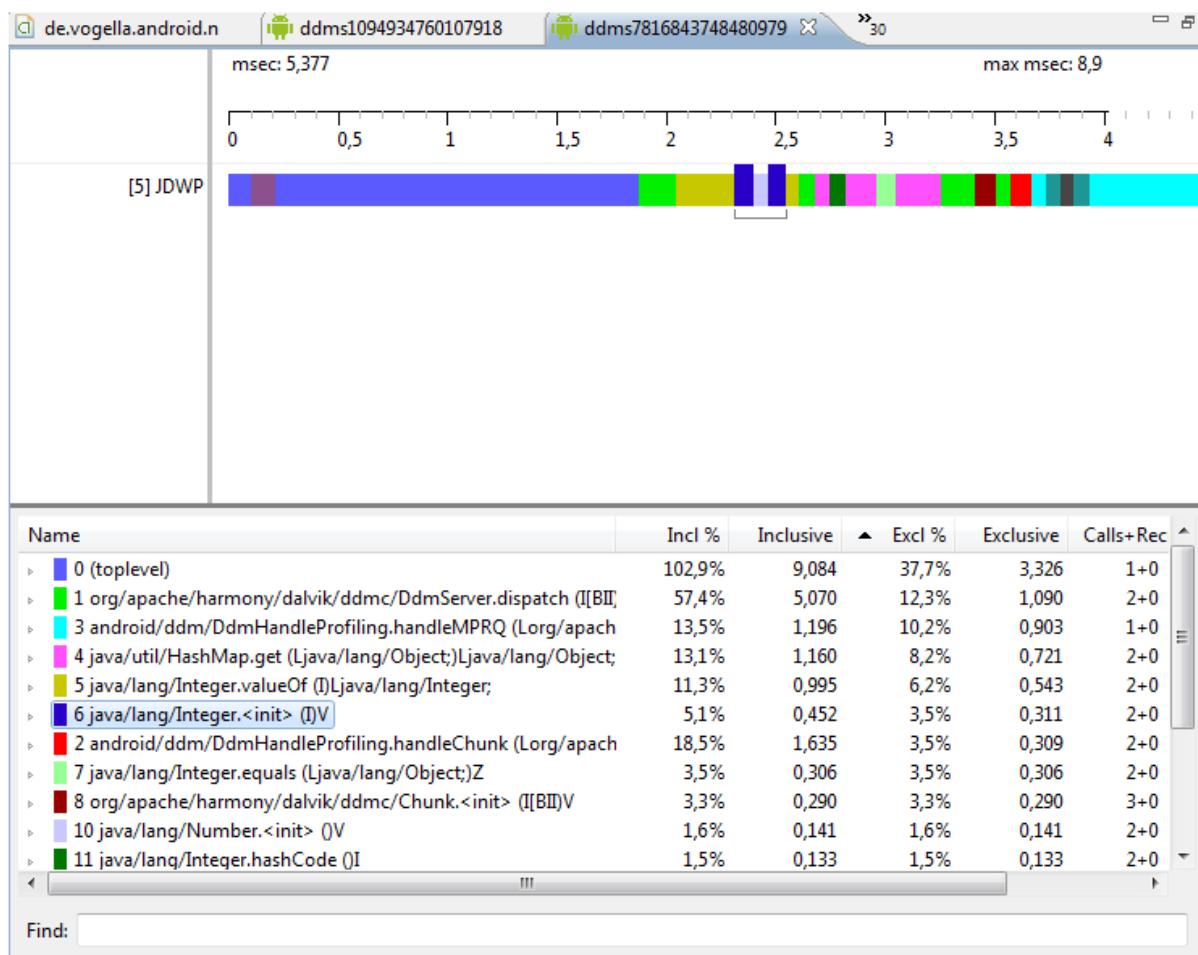
محیط برنامه نویسی به شما امکان می‌دهد تا از طریق ابزار Android Device Monitor روند اجرای اپلیکیشن را رصد و مشاهده (trace) نمایید. برای این منظور کافی است مسیر رو به روی را طی نمایید:

.Tools ▶ Android ▶ Android Device Monitor

جهت فعال سازی قابلیت رد گیری روند اجرای اپلیکیشن، ابتدا فرایند اپلیکیشن خود را در کادر Devices انتخاب نموده و سپس دکمه‌ی Start Method Profiling را مطابق تصویر زیر کلیک کنید.



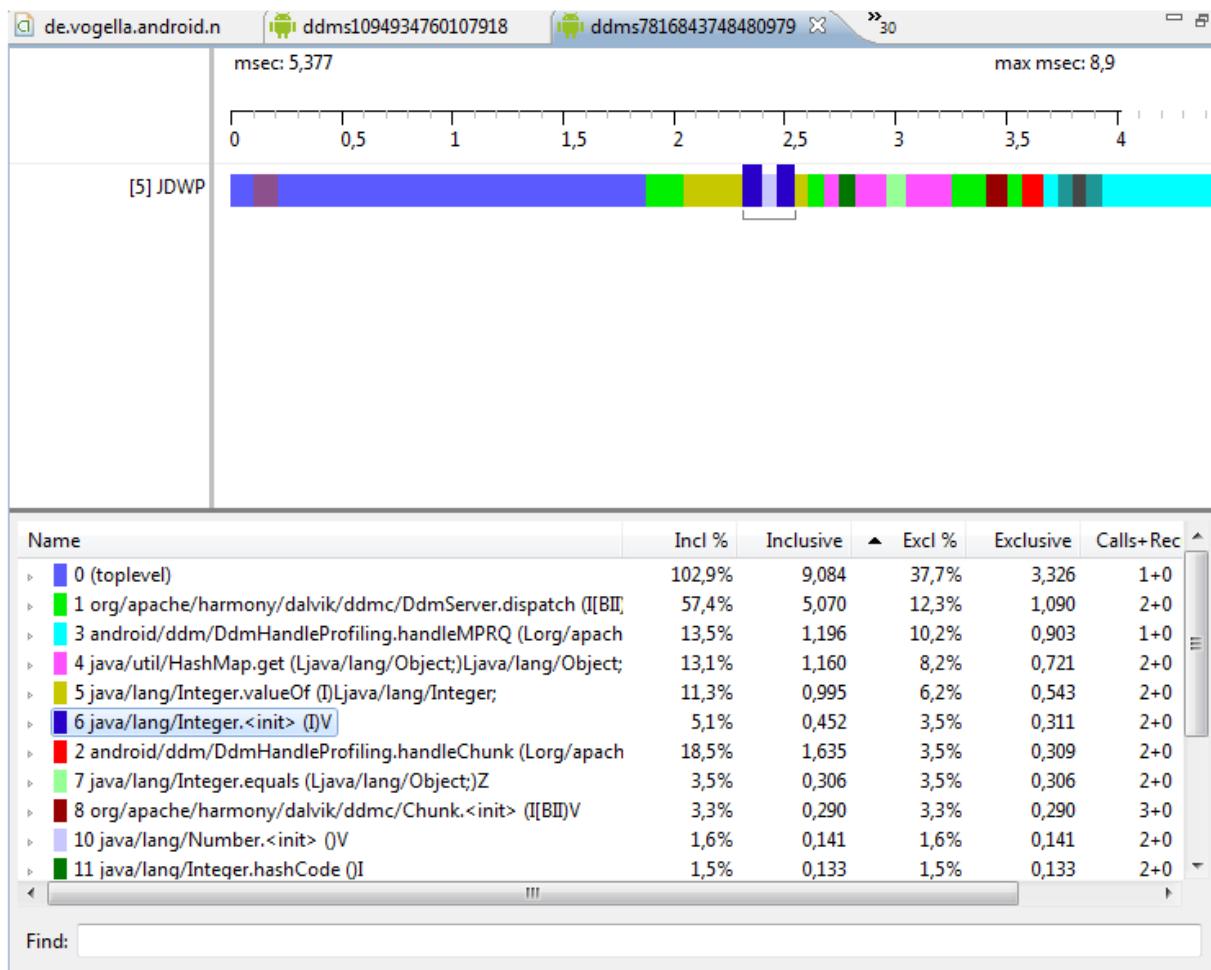
اپلیکیشن را اجرا نموده و مجدداً دکمه‌ای که جهت فعال سازی ردگیری جریان اجرای اپلیکیشن کلیک کردید، کلیک نمایید. در پی این اقدام، یک ویرایشگر جدید باز شده، که نتایج ردگیری و ضبط شده از روند اجرای اپلیکیشن را به نمایش می‌گذارد.



796

آدرس آموزشگاه: تهران - خیابان شریعتی - بالاتر از خیابان ملک - جنب بانک صادرات - پلاک 651 طبقه دوم - واحد 7

88146323 - 88446780 - 88146330



می توانید تصویر را بزرگنمایی کرده و جزئیات را دقیق تر مشاهده نمایید. برای کوچک نمایی، کافی است بر روی خط نمایشگر زمان دوبار کلیک کنید.

### 3-50-7- راه اندازی و فراخوانی ابزار TraceView از طریق خط دستور (command line)

برای اینکه پروسه‌ی رصد اجرای کد برنامه (trace) آغاز شود، کافی است تکه کد زیر را بین دو دستور زیر درج نمایید.

```
android.os.Debug.startMethodTracing("yourstring");
// ... your code is here
android.os.Debug.stopMethodTracing();
```

پارامتر "yourstring" به سیستم اندروید اعلان می کند که بایستی داده ها را تحت آدرس "/sdcard/yourstring.trace" ذخیره نماید. جهت جایگذاری داده ها در حافظه ی خارجی XML، اپلیکیشن شما می بایست مجوز WRITE\_EXTERNAL\_STORAGE را در لایه ی (sdcard) تعریف کرده باشد. پس از اجرای اپلیکیشن، می توانید نتایج را با اجرای دستور زیر در پنجره ی فرمان adb (command line tool)، در مسیر دلخواه کپی کنید.

```
adb pull /sdcard/yourstring.trace
traceview yourstring
```

با فراخوانی این دستور در پنجره ی فرمان adb، ابزار Traceview فعال شده و به شما اجازه می دهد تا داده های مربوط به کارایی اپلیکیشن خود را به صورت دیداری (visual) و قالب اطلاعات قابل مشاهده در نمایشگر بررسی و دنبال نمایید. در کادر DDMS یک دکمه ی trace و ردگیری جریان اپلیکیشن وجود دارد. با استفاده از این دکمه، جریان اجرای اپلیکیشن در حین اجرا و پیشرفت، رصد شده و لزومی به تنظیم مجوز خاصی در لایه ی XML نیست.

## تمرین: استفاده ی کاربردی از Traceview

### استفاده از ابزار Traceview در پروژه

یک اپلیکیشن اندرویدی جدید با نام پکیج com.vogella.android.traceview (top level package) ایجاد نمایید.

کلید زیر را به فایل values/strings.xml اضافه نمایید.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Traceview Example</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>
    <string name="number_template"><b>Random number: %1$s</b></string>
</resources>
```

یک فایل layout با نام rowlayout.xml و محتوای زیر ایجاد نمایید.

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="?android:attr/listPreferredItemHeight"
    android:orientation="horizontal" >
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical" >
        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="vertical" >
            <TextView
                android:id="@+id/textView1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Entry"
                android:textAppearance="?android:attr/textAppearanceListItem" />
            <TextView
                android:id="@+id/textView2"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Stub"
                android:textAppearance="?android:attr/textAppearanceListItemSmall" />
        </LinearLayout>
    </LinearLayout>
    <ImageView
        android:id="@+id/imageView1"
        android:layout_width="10dp"
        android:layout_height="?android:attr/listPreferredItemHeight"
        android:background="@android:color/black" />
    <TextView
        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:text="Medium Text"
        android:textAppearance="?android:attr/textAppearanceMedium" />
</LinearLayout>

```

یک کلاس adapter برای آبجکت ListView اپلیکیشن خود به صورت زیر پیاده سازی نمایید.

```

package com.vogella.android.traceview;
import java.util.Collections;
import java.util.List;
import android.content.Context;
import android.content.res.Resources;
import android.os.Debug;
import android.text.Html;
import android.view.LayoutInflater;

```

```

import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.TextView;
public class MyArrayAdapter extends ArrayAdapter<String> {
    private List<String> values;
    private Context context;
    public MyArrayAdapter(Context context, List<String> values) {
        super(context, R.layout.rowlayout);
        this.context = context;
        this.values = values;
    }
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        Debug.startMethodTracing("getViewOfTrace");
        // Ensure sorted values
        Collections.sort(values);
        LayoutInflater inflater = (LayoutInflater) context
            .getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        View view = inflater.inflate(R.layout.rowlayout, parent, false);
        Resources res = context.getResources();
        String text = String.format(res.getString(R.string.number_template),
            values.get(position));
        CharSequence styledText = Html.fromHtml(text);
        TextView textView = (TextView) view.findViewById(R.id.textView3);
        textView.setText(styledText);
        Debug.stopMethodTracing();
        return view;
    }
}

```

حال یک آبجکت ListView در کلاس activity (جهت ساخت یک لیست ساده) خود که 1000 رشته با مقدار تصادفی را در صفحه به نمایش می گذارد، پیاده سازی کنید.

```

package com.vogella.android.traceview;
import java.math.BigInteger;
import java.security.SecureRandom;
import java.util.ArrayList;
import java.util.List;
import android.app.ListActivity;
import android.os.Bundle;
public class MainActivity extends ListActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        List<String> list = createValues();
        MyArrayAdapter adapter = new MyArrayAdapter(this, list);
        setListAdapter(adapter);
    }
}

```

```

private static List<String> createValues() {
    SecureRandom random = new SecureRandom();
    List<String> list = new ArrayList<String>();
    for (int i = 0; i < 1000; i++) {
        String string = new BigInteger(130, random).toString(32);
        list.add(string);
    }
    return list;
}

```

مجوز لازم برای ذخیره و درج اطلاعات در حافظه‌ی خارجی دستگاه خود را در لایه‌ی XML اپلیکیشن اعلان نمایید.

#### 7-50-4- رصد و سنجش کارایی اپلیکیشن (Trace)

اپلیکیشن خود را اجرا کنید. سپس از طریق adb به دستگاه مربوطه وصل شده و اطلاعات مربوط به رصد کارایی اپلیکیشن را در حافظه جایگذاری نمایید. حال خروجی مربوط به کارایی اپلیکیشن را تجزیه و تحلیل کنید.

#### 7-50-5- برطرف کردن کاستی‌های مربوط به کارایی اپلیکیشن

اکنون زمان آن فرا رسیده تا کارایی اپلیکیشن را بر اساس اطلاعات و خروجی Traceview افزایش داده و کاستی‌های آن را در جهت بهبود تجربه‌ی کاربری برطرف نمایید.

- مقدار attribute ای که ظاهر متن را در متد `getView()` تعیین می‌کند (مقدار متن را بر روی `bold` یا برجسته تنظیم می‌کند)، با دستور `android:textStyle="bold"` متناظر در لایه‌ی XML، داخل فایل LAYOUT جایگزین نمایید تا بدین وسیله از فراخوانی متد `Html.fromHtml()` جلوگیری شود.
- مرتب سازی را به مکان دیگری انتقال دهید.
- چنانچه مقدار `convertView` برابر `null` نبود، از آن بار دیگر در کلاس Adapter استفاده نمایید.

- با پیاده سازی الگو توسعه ی ListView در آبجکت HolderPattern، از فراخوانده شدن متند `findViewById()` جلوگیری نمایید.

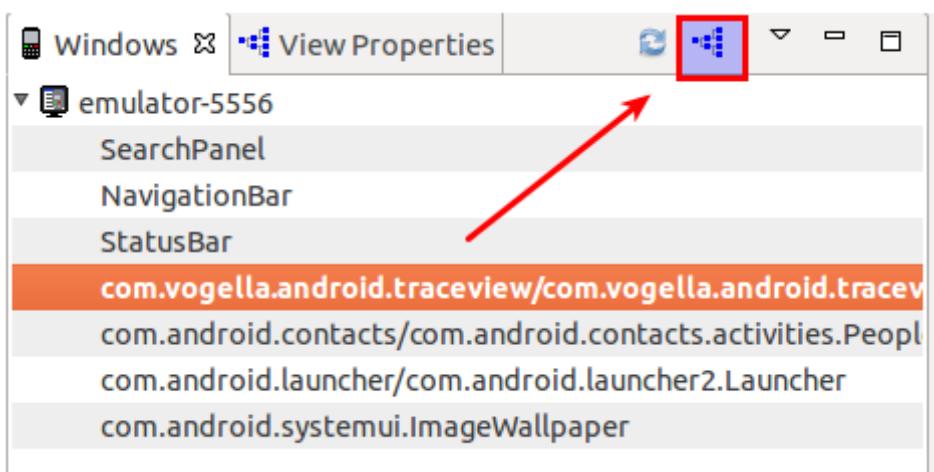
**توجه:** layout به طور بهینه طراحی نشده است. مثال فوق را در تمرین HierarchyView مجدداً بکار خواهیم برد.

## (ابزار نمایش درختی view ها) Hierarchy Viewer-51-7

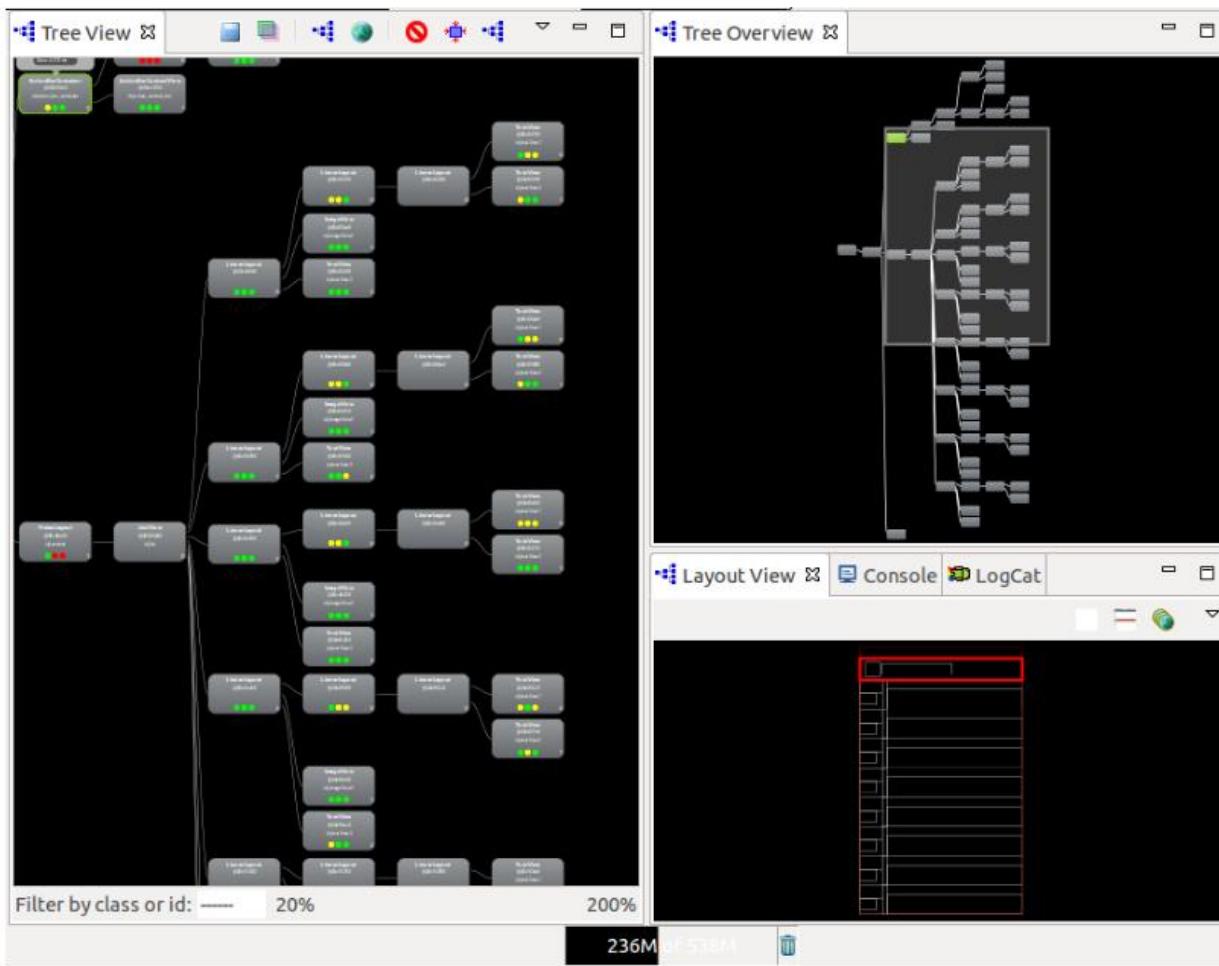
Hierarchy View به شما این امکان را می دهد تا نمودار درختی و سلسله مراتبی View اپلیکیشن اندروید خود را به صورت دیداری در محیط کاری اندروید ترسیم کرده و سپس لایه های غیرضروری و اضافی را در نمودار درختی مزبور شناسایی نمایید.

برای دسترسی به Hierarchy View، کافی است مسیر رو به رو را طی نمایید: Window > Open > Perspective > Other... > Hierarchy View

در view یا کادر Widows می توانید فرایندی که می خواهید view hierarchy برای آن مورد تحلیل قرار گیرد را انتخاب نمایید.



layout ای که در حال حاضر فعال می باشد، مورد تحلیل قرار گرفته و به نمایش گذاشته می شود.



کادری که Tree View نام دارد، در هر یک از view ها سه دایره‌ی کوچک رنگی به نمایش می‌گذارد. اولین دایره، مدت زمانی که برای محاسبه‌ی اندازه‌ی view لازم است را بیان می‌کند. دومین دایره بیانگر زمان لازم برای ایجاد layout و طرح کلی می‌باشد و سومین دایره مدت زمان لازم برای ترسیم view را نشان می‌دهد. کارهای هزینه بر و طولانی با رنگ زرد یا قرمز برای برنامه نویس نمایش داده می‌شود.

## تمرین: استفاده‌ی کاربردی از ابزار Hierarchy Viewer

در تمرین جاری، همچنان از مثال قبلی که در تمرین Traceview بکار برده شد، استفاده خواهیم کرد. Hierarchy View را باز کرده و لایه‌ی view را دقیق مورد بررسی قرار دهید.

گرچه هیچ یک از لایه‌های layout، با توجه به دایره‌ی رنگی داخل view‌ها، از نظر منابع مورد استفاده و زمان اجرا هزینه بر و طولانی نیستند، با این وجود لایه‌های view‌ها غیرضروری در آن به چشم می‌خورند. حال لایه‌های غیرضروری را حذف نمایید.

## 52-بهینه سازی Layout

اکنون زمان آن فرا رسیده که فایل layout تنظیم کننده ظاهر و طرح کلی اپلیکیشن خود را بهینه نمایید. تصویر زیر با مثال نمایش می‌دهد چگونه با استفاده از Framelayout و دو المان رابط کاربری TextView و ImageView نتیجه‌ی ارائه شده را تولید نمایید. همان‌طور که در تصویر حاضر مشاهده می‌کنید، کامپوننت رابط کاربری TextView در layout جایگذاری شده و محتوای متنی آن به دو روش متفاوت سبک یا style دهی شده‌اند.



یک پروژه‌ی جدید به نام `com.vogella.android.textview.spannablestring` ایجاد نمایید.  
حال دو سبک دهی (styles) جدید به محتوای فایل `styles.xml` اضافه نمایید.

```
<resources xmlns:android="http://schemas.android.com/apk/res/android">
<!--
Base application theme, dependent on API level. This theme is replaced
by AppBaseTheme from res/values-vXX/styles.xml on newer devices.
-->
<style name="AppBaseTheme" parent="android:Theme.Light">
<!--
Theme customizations available in newer API levels can go in
res/values-vXX/styles.xml, while customizations related to
backward-compatibility can go here.
-->
</style>
<!-- Application theme. -->
<style name="AppTheme" parent="AppBaseTheme">
<!-- All customizations that are NOT specific to a API-level are here. -->
</style>
<style name="textHeader">
<item name="android:padding">4dip</item>
<item name="android:textAppearance">?android:attr/textAppearanceLarge</item>
```

```

<item name="android:textColor">#000000</item>
<item name="android:fontFamily">sans-serif-condensed</item>
</style>
<style name="textbody">
<item name="android:padding">4dip</item>
<item name="android:textAppearance"><?android:attr/textAppearanceSmall</item>
<item name="android:textSize">16sp</item>
<item name="android:textColor">#c0c0c0</item>
</style>
</resources>

```

یک فایل layout با محتوای زیر ایجاد نمایید.

```

<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/FrameLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >
    <TextView
        android:id="@+id/input"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom"
        android:layout_margin="16dp"
        android:text="@string/hello_world"
        android:textSize="32sp" />
    <ImageView
        android:id="@+id/imageView1"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:contentDescription="image"
        android:src="@drawable/vogella" />
</FrameLayout>

```

متن موجود در المان TextView خود را، همان طور که در کد زیر نمایش داده شده، با دو نمونه  
ی مجزا از TextAppearanceSpan سبک دهی و تنظیم (style) نمایید.

```

package com.vogella.android.textview.spannablestring;
import android.app.Activity;
import android.os.Bundle;
import android.text.Spannable;
import android.text.SpannableString;
import android.text.style.TextAppearanceSpan;
import android.view.Menu;
import android.widget.TextView;
public class MainActivity extends Activity {

```

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    TextView textView = (TextView) findViewById(R.id.input);
    String header = "This is the header";
    String description = "This is the description";
    Spannable styledText = new SpannableString(header + "\n" + description);
    TextAppearanceSpan span1 = new TextAppearanceSpan(this,
        R.style.textHeader);
    TextAppearanceSpan span2 = new TextAppearanceSpan(this,
        R.style.textbody);
    styledText.setSpan(span1, 0, header.length(),
        Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
    styledText.setSpan(span2, header.length() + 1, header.length() + 1
        + description.length(), Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
    textView.setText(styledText);
}
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.main, menu);
    return true;
}
}

```

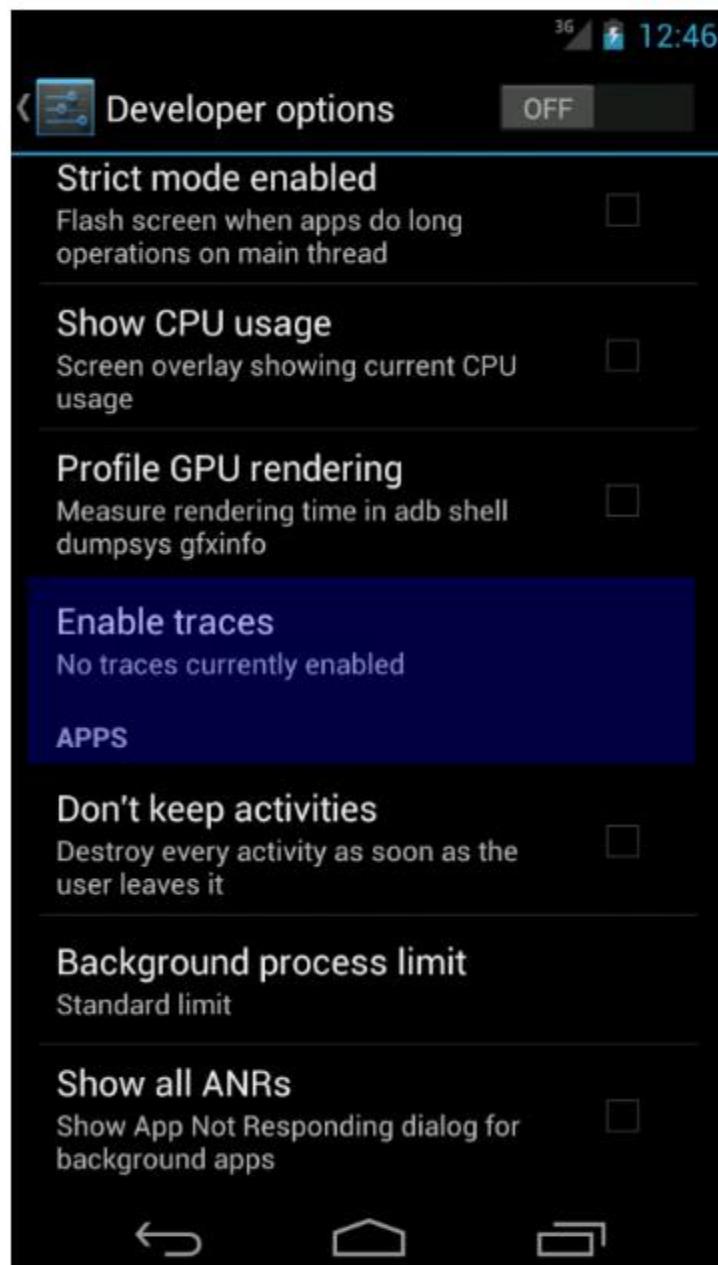
در نهایت مشاهده خواهید کرد که layout جدید بسیار سریع تر از layout قبلی که مبتنی بر RelativeLayout می باشد، عمل می کند. لازم به توضیح است که در layout جاری از HTML خودداری شده چرا که استفاده از مفسر با تبدیلیگر HTML parser بسیار هزینه بر و سنگین می باشد.

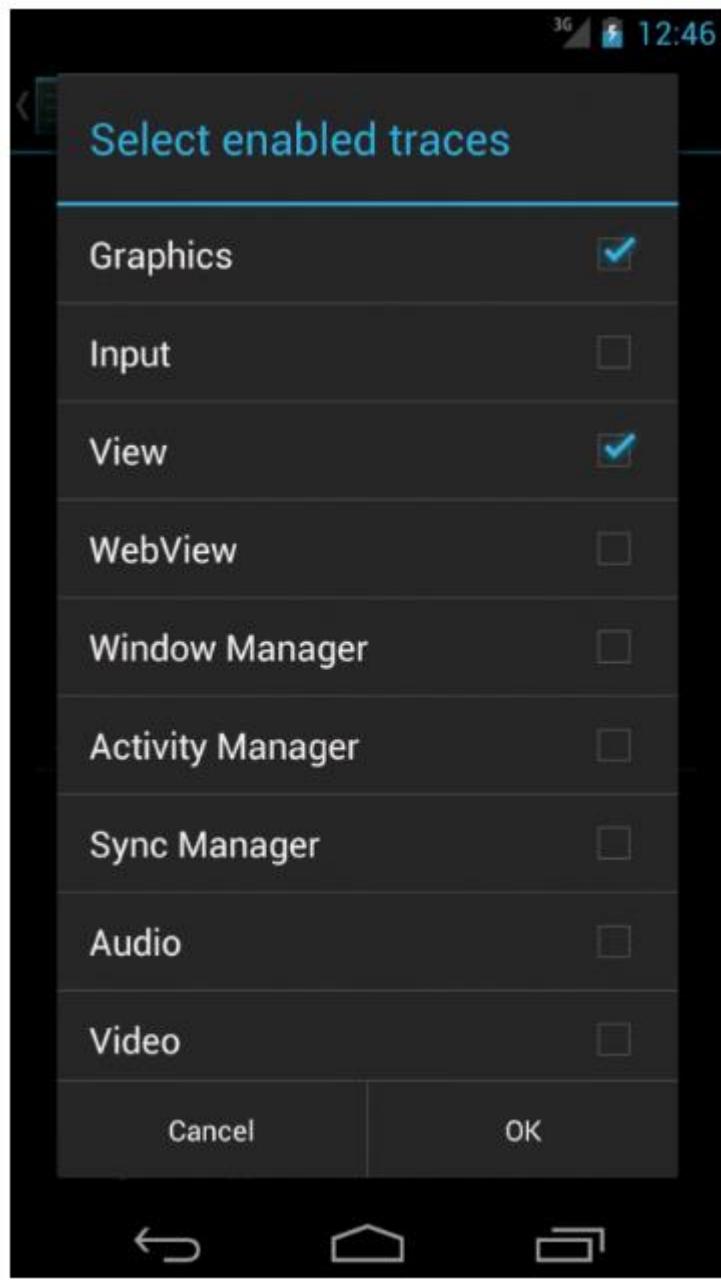
## 7-53-تهیه ی روگرفت از حافظه (Memory Dumps)

در صورت تمایل می توانید یک روگرفت یا تصویر لحظه ای از حافظه تهیه کرده و سپس آن را به واسطه ی ابزار Memory Analyzer Eclipse محیط کاری بررسی و تحلیل نمایید.

## 7-54-ابزار Systrace

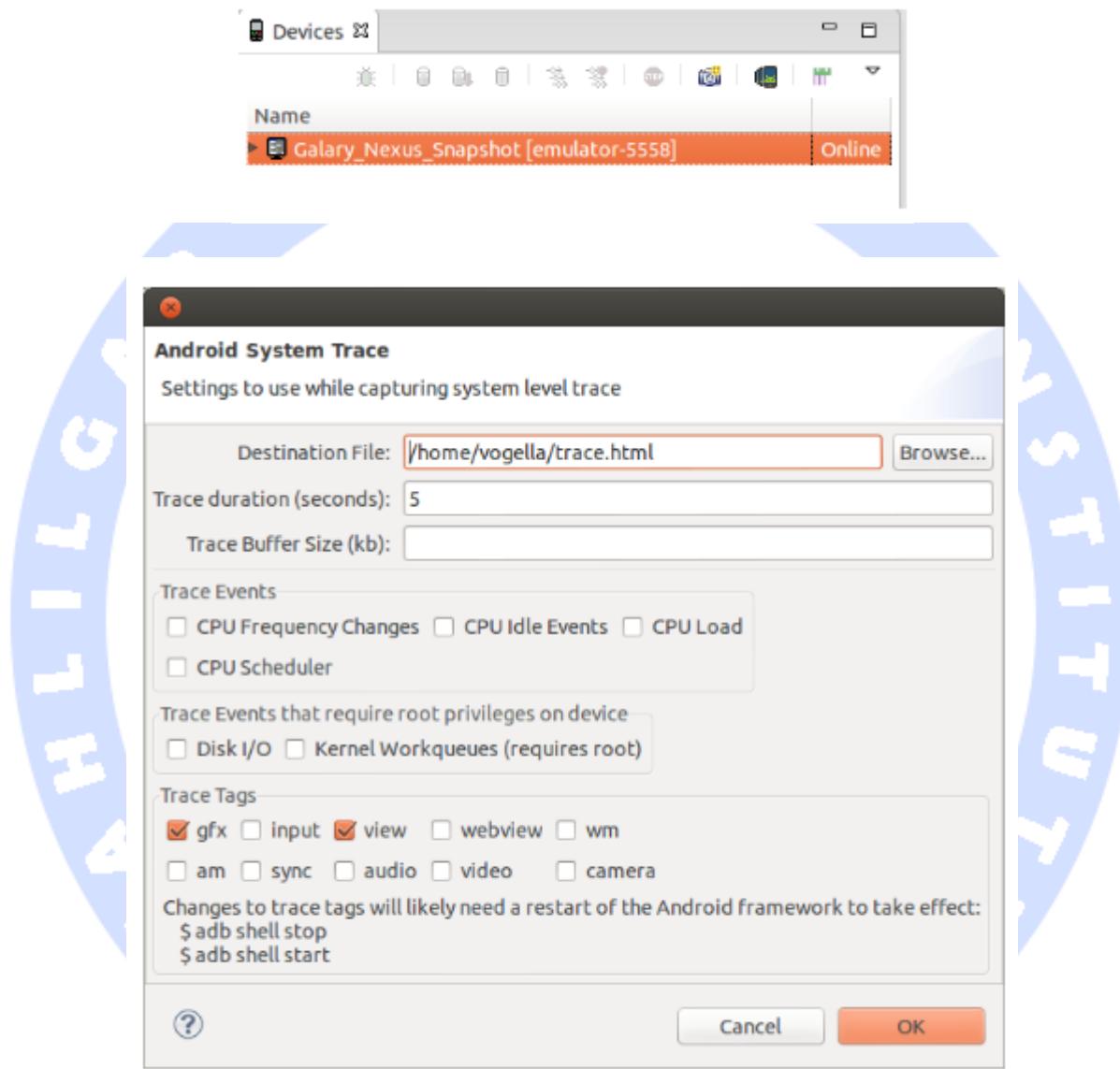
به شما این امکان را می دهد تا کارایی اپلیکیشن را خود به طور مسقیم در سطح هسته سیستم (kernel level) بررسی نمایید. جهت فعال سازی این ابزار، پس از انتخاب Developer options، المان Enable traces را کلیک نمایید. در کادر محاوره‌ی dialog (بعدی، می توانید به طور صریح مشخص نمایید از کدام event ها، به عنوان مثال، بایستی گزارش تحلیلی و جزئی تهیه شود profile).





به منظور استفاده از ابزار `systrace.py`, یک پنجره فرمان (terminal) باز کرده، سپس `systrace.py` را از پوشه اصلی (`android_sdk_installdir/tools`) اجرا نمایید. ممکن است لازم باشد آن را بر روی یک فایل اجرایی (executable) تنظیم نمایید (در لینوکس `chmod +x systrace.py`).

در صورت تمایل می توانید ابزار مزبور را از طریق DDMS مسقیماً از محیط کاری Eclipse اجرا نمایید.



ابزار Systrace رخدادها را به مدت پنج ثانیه ضبط (capture) می کند. در نتیجه، ابزار نام بردۀ یک فایل HTML ایجاد کرده و به شما این امکان را می دهد تا مشکلات احتمالی را تجزیه و تحلیل نمایید.

## 7-55- شبیه سازی چگالی یا تراکم پیکسلی (pixel density) از طریق خط دستور

شما می توانید با استفاده از خط دستور (command line)، چگالی پیکسلی و وضوح تصویر دستگاه های مختلف اندروید را شبیه سازی کنید. این امر به شما اجازه می دهد تا از یک دستگاه استفاده کرده و به وسیله‌ی آن، چگالی پیکسلی و وضوح تصویر دیگر دستگاه های اندروید را شبیه سازی نمایید.

```
// Set the display size
adb shell am display-size 600x800
// Set the display density
adb shell am display-density 80
```

## 7-56- قالب های آماده یا الگوهای پیاده سازی پروژه‌ی اندروید (Android template)

شما می توانید قالب های آماده دلخواه خود را برای ویژاد ساخت پروژه‌ی اندروید (Android project creation wizard) تعریف نمایید. جهت دستیابی به اطلاعات بیشتر در این زمینه می توانید به لینک زیر مراجعه نمایید:

<https://plus.google.com/113735310430199015092/posts/XTKTAmk4As8>

قالب و الگوهای بیشتر کدنویسی تحت آدرس <https://github.com/Jgilfelt/android-adt-templates> قابل دسترسی می باشد.

## 7-57- گزارش گیری و ثبت اطلاعات مربوط به کارایی موتور گرافیکی (ابزار GPU rendering)

در بخش **Developer Options**، داخل **Settings** یا در دستگاه واقعی اندروید خود، این امکان برای شما وجود دارد که ابزار **Profile GPU rendering** را فعال نموده و مدت زمان مورد نیاز برای ترسیم 128 فریم توسط سیستم را رصد و ثبت نمایید.

پس از فعال سازی این امکان و راه اندازی مجدد اپلیکیشن، می توانید با فراخوانی دستور زیر در پنجره **cmd** فرمان، اطلاعات مربوطه را واکشی نمایید.

```
adb shell dumpsys gfxinfo your_package
```

برای مثال جهت سنجش **frame rate** (سرعت پخش فریم ها) اپلیکیشن با پیمایش به سمت پایین در یک لیست. در بیشتر موارد، لازم است با اپلیکیشن خود تعامل کنید تا مجبور شود خود را مجدداً ترسیم کند.

در گزارش نهايی که در اختیار شما قرار می گيرد، بخشی به نام **Profile data in ms** را پیدا کنيد.

## 58-7-تحلیل و بررسی وضعیت Overdraw (ترسیم یک آیتم بر روی آیتم دیگر)

زمانی رخ می دهد که دو آیتم همپوشانی داشته باشند، بدین معنی یک آیتم بر روی آیتم دیگری رسم شود. به طور مثال، می توان به زمانی اشاره کرد که پس زمینه **Window** activity، یک **TextView** باشد. حال اگر المان **Window** به اپلیکیشن اضافه شود، این المان طبیعتاً بر روی **Window** ترسیم شده و قرار می گیرد.

بنابراین وضعیت **overdraw** یک امر طبیعی بوده و رخداد آن مورد انتظار می باشد. با این حال توصیه می شود جهت بالا بردن کارایی اپلیکیشن و تجربه ی کاربری، از ترسیم آبجکت های متعدد بر روی هم و رخداد **overdraw** به صورت بیش از حد خودداری شود.

دلیل اصلی رخداد **overdraw** بی رویه می تواند **view hierarchies** و نمودار درختی بسیار پیچیده ی **view** ها باشد. به طور کلی وضعیت **2x overdraw** (زمانی که یک پیکسل سه بار ترسیم می شود) یک امر طبیعی است اما بیشتر از آن می بایست اجتناب شود چرا که افت کارایی را به دنبال دارد.

می توانید از طریق Show GPU overdraw و فعال گردن گزینه **i** Development Settings وضعیت رخداده **i** overdraw و view های ترسیم شده بر روی هم را به صورت دیداری در نمایشگر داشته باشید. این تنظیم بر اساس overdraw ها، رنگ به صفحه نمایش تخصیص می دهد. جدول زیر مجموعه رنگ بکار رفته را همراه با معنی هر یک شرح می دهد.

رنگ مربوطه	معنی رنگ
-	هیچ view ای بر روی view دیگر ترسیم نشده. به عبارت دیگر، وضعیت overdraw رخ نداده است.
آبی	یک پیکسل دو بار رسم شده است. $= 1x$ overdraw
سبز	$= 2x$
قرمز روشن	یک پیکسل سه بار رسم شده و می تواند کمی مشکل زا باشد. اما با این وجود، نواحی کوچک قرمز هنوز مشکل جدی ایجاد نمی کند.
قرمز تیره	یک پیکسل ۵ بار یا حتی بیشتر ترسیم شده است. این رنگ نشانگر بروز مشکل می باشد.

پس از دریافت نسخه **i** دیداری و بررسی بخش هایی که مشکل زا بودند، می توانید نمودار درختی View را با ابزار Hierarchy Viewer تحلیل نمایید.

## 7-59-ابزار رایگان و متن باز کارا برای تحلیل و بهینه سازی کارایی اپلیکیشن

## 7-59-1 استفاده از Leak Canary برای یافت و برطرف نمودن هدر رفت حافظه (memory leak)

یک ابزار متن باز است که می‌توان با اپلیکیشن اندروید ادغام نموده و به کمک آن هدر رفت حافظه را به صورت خودکار شناسایی کرد. استفاده از ابزار نام برده بسیار آسان است و شما می‌توانید آن را به عنوان کتابخانه (dependency) به فایل Gradle build اضافه نموده و سپس کتابخانه‌ی مذبور را در کلاس application خود مقداردهی اولیه (initialize) نمایید. با این کار یک lifecycle listener و گوش فراخوان به چرخه‌ی حیات ها اضافه شده و دقیقاً بررسی می‌کند آیا متد destroy آن‌ها در زمان مناسب فراخوانی می‌شود یا خیر.

لازم به توضیح است که Canary leak به علت open source بودن به صورت مداوم ویرایش و بروز رسانی می‌شود. جهت اطلاع از نحوه‌ی بکارگیری این ابزار به آدرس Leak Canarys Github، صفحه‌ی <https://github.com/square/leakcanary> مراجعه نمایید.

## 7-59-2 استفاده از AndroidDevMetrics جهت مشاهده اطلاعات مربوط به کارایی اپلیکیشن

ابزار AndroidDevMetrics به توسعه دهنده‌ی این امکان را می‌دهد تا سرعت اجرای عملیات معمول همچون مقداردهی اولیه‌ی آبجکت یا فراخوانی توابع مدیریت چرخه‌ی حیات (lifecycle) (از جمله ()onResume() و ()onStart().onCreate()) را تحلیل نماید.

همان طور که درباره ای ابزار قبلی گفته شد، یک پروژه ای متن باز (Open Source) است. بدین معنی که ممکن است به صورت مداوم بروز رسانی شده و مورد ویرایش قرار گیرد. برای اطلاع از نحوه ای استفاده از این ابزار متن باز می توانید به آدرس [AndroidDevMetrics](https://github.com/frogermcs/androiddevmetrics) - صفحه ای مراجعه نمایید.





816

آدرس آموزشگاه : تهران - خیابان شریعتی - بالاتر از خیابان ملک - جنب بانک صادرات - پلاک 651 طبقه دوم - واحد 7

88146323 - 88446780 - 88146330

## بخش اول :

# کامپایل / Build پروژه های اندرویدی با سیستم

آموزش حاضر نحوه‌ی استفاده از سیستم Gradle برای کامپایل پروژه‌های اندرویدی را شرح می‌دهد. در این مبحث با نحوه‌ی تنظیم نسخه‌های مختلف flavor (flavor) آشنا خواهید شد.

## 1-8-استفاده از Gradle برای کامپایل پروژه‌های اندرویدی

### 1-8-1-فایل‌های build اپلیکیشن‌های اندروید

فرایند build پروژه‌های اندرویدی توسط سیستم کامپایل Gradle مدیریت می‌شود. زمانی که پروژه‌ی جدیدی در محیط برنامه نویسی اندروید ایجاد می‌کنید، به دنبال آن build script (فایل build) نیز به طور خودکار تولید می‌شوند. در واقع محیط کاری Android Studio به صورت خودکار runtime و بستر اجرای Gradle را دربرمی‌گیرد، از این رو احتیاجی به نصب ابزار یا افزونه‌ی جداگانه وجود ندارد.

سیستم Gradle طوری طراحی شده که از سناریوهای پیچیده مختلف برای ایجاد اپلیکیشن‌های اندرویدی پشتیبانی کند:

- Multi-distribution: وضعیتی که یک اپلیکیشن بایستی برای چندین سرویس گیرنده یا شرکت اختصاصی تنظیم شود.
- Multi-apk: تولید چندین فایل apk از یک اپلیکیشن جهت سازگاری با انواع دستگاه‌ها و استفاده‌ی مجدد از بخش‌هایی از کد برنامه در این فایل‌های apk.

همچنین می‌توانید از یک wrapper script که سیستم Gradle تولید می‌کند، استفاده نمایید یک اسکریپت است که چندین دستور و اسکریپت که قابلیت wrapper script (wrapper script)

اجرا در برنامه‌ی اصلی را ندارد، در بر می‌گیرد). این wrapper به شما امکان می‌دهد تا یک build (command line) را بدون اینکه لازم باشد هیچ ابزار و فایل دیگری نصب کنید، از خط دستور (build) اجرا نمایید.

زمانی که بر روی دکمه‌ی run در محیط کاری Android Studio کلیک می‌کنید، تسلیم مربوطه‌ی زمانی که راه اندازی شده و اپلیکیشن اجرا می‌شود.

**نکته:** می‌توانید به ورژن‌های مختلف افزونه‌ی Gradle تحت آدرس <https://jcenter.bintray.com/com/android/tools/build/gradle/> دسترسی داشته باشید.

**۱-۸-پروسه‌ای که کد برای تبدیل به اپلیکیشن اندروید طی می‌کند**

کامپایلر جاوا فایل‌های حاوی کدهای جاوا (java source file) را به فایل‌هایی با پسوند .class تبدیل می‌کند (فایل‌هایی که در برداشته‌ی bytecode ها و کدهای زبان میانی بوده و بر روی JVM اجرا می‌شود). Android SDK یک ابزار به نام dx دارد که فایل‌های .class را به فایل‌هایی با پسوند dex. (فایل‌های اجرایی Dalvik) تبدیل می‌کند. تمامی فایل‌های کلاس اپلیکیشن در فایل اجرایی Dalvik جایگذاری می‌شوند. در طی این پروسه تبدیل، اطلاعات مزاد موجود در فایل‌های class. داخل فایل dex. بهینه می‌شوند. برای مثال، چنانچه یک String در چندین فایل class. وجود داشته باشد، فایل dex. تنها یک reference یا اشاره گر به متغیر مزبور را نزد خود نگه می‌دارد.

فایل‌های dex. از نظر حجم بسیار سبک‌تر از فایل‌های class. متناظر هستند.

فایل dex. و سایر منابع و محتوا همچون image و فایل‌های XML همگی در قالب یک فایل apk گنجانده می‌شوند. این وظیفه را ابزار پکیج بندی منابع و محتوا اپلیکیشن‌های اندرویدی به نام (Android Asset Packaging Tool) aapt بر عهده دارد.

فایل apk. خروجی، دربردارنده ی تمامی داده های لازم برای اجرای اپلیکیشن اندروید بوده و به راحتی از طریق ابزار adb قابل نصب بر روی دستگاه اندرویدی می باشد.

از ویرایش 5.0 به بعد، Android RunTime به عنوان ابزار بارگذاری و اجرا / runtime تمامی اپلیکیشن های اندرویدی ایفای نقش می کند. ART یک قابلیت است که اپلیکیشن را درست به مجرد درخواست کاربر لود کرده و اجرا می کند. بدین وسیله سرعت اجرای اپلیکیشن به مراتب نسبت به زمانی که از Dalvik استفاده می شد، بالا می رود. در واقع ART از تلفیقی از دو روش کامپایل Ahead of Time (ترجمه ی کد زودتر از درخواست کاربر) و Just-in-Time (ترجمه ی کد به محض درخواست کاربر) استفاده می کند. به هنگام نصب یک اپلیکیشن بر روی دستگاه اندرویدی، کد اپلیکیشن به زبان ماشین ترجمه می شود.

ابزار dex2oat فایل dex. که خروجی مجموعه ابزار اندروید می باشد را تحويل گرفته و آن را به یک فایل (EFL) با فرمت Executable and Linkable تبدیل می کند. این فایل حاوی کد dex، کد کامپایل شده برای سخت افزار و محیط جاوا / native code و meta-data می باشد.

## راه اندازی Gradle از خط دستور (command line)

می توانید فایل build خود را از خط دستور اجرا کنید. برای این منظور، دستور زیر را از پوشه ی اصلی پروژه (main project directory) اجرا نمایید. لازم است Gradle بر روی دستگاه مربوطه نصب شده باشد و یا کد دربرگیرنده ی Gradle wrapper gradlew (Gradle wrapper) را برای اجرای build بکار ببرید. در صورت لزوم wrapper خود به صورت خودکار سیستم Gradle را دانلود می کند. در زیر لیست task های مهم Android Gradle را با شرح کاربرد مشاهده می شود:

دستور	شرح کاربرد
./gradlew build	پروژه را build و کامپایل نموده و هر دو تست <code>check</code> و <code>assemble</code> را اجرا می کند.
./gradlew clean build	پروژه را کامل از صفر کامپایل و build می کند.
./gradlew clean build	پروژه را از صفر build و کامپایل می کند.
./gradlew test	تست ها را اجرا می کند.
./gradlew connectedAndroidTest	تست های مبتنی بر instrumentation را اجرا می کند.

جهت مشاهده تمامی task های موجود، دستور `gradle wrapper` را فراخوانی کنید.

```
gradle build
# alternatively speedup second gradle build by holding it in memory
# gradle build --daemon
```

با اجرای این دستور، یک پوشه `build` برای قرارگیری خروجی Gradle build ایجاد می شود. به صورت پیش فرض، `Gradle build` دو فایل با پسوند `.apk` در پوشه `build/outputs/apk` ایجاد می کند.

به منظور `build` و راه اندازی تست های نرم افزاری `unit test` خود بر روی دستگاه مجازی جاوا (JVM)، دستور زیر را فراخوانی کنید.

```
gradle test
```

حال جهت کامپایل، `build` و راه اندازی تست های مبتنی بر `instrumentation` خود بر روی دستگاه واقعی اندروید، دستور زیر را اجرا نمایید.

```
gradle connectedCheck
```

## 8-1-3 resource shrinking

سیستم کامپایل و build پروژه Gradle، قادر است با استفاده از روش بهینه سازی منابع (resource shrinking) در زمان کامپایل و build پروژه، کلاس ها و منابع غیرضروری جاوا را حذف نماید. بدین معنی که منابع بلااستفاده از پکیج اپلیکیشن به صورت خودکار حذف می شوند. علاوه بر آن، تمامی منابع غیرضروری از کتابخانه هایی که مورد استفاده ی پروژه هستند، حذف شده حجم نرم افزار به طور قابل توجهی کاهش می یابد.

به منظور فعال سازی قابلیت بیهنه سازی منابع (resource shrinking)، محتوای فایل build خود را به صورت زیر ویرایش کنید.

```
android {
    ...
    buildTypes {
        release {
            minifyEnabled true
            shrinkResources true
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}
```

## 8-1-4 تعریف dependency ها و اعلان ورژن کتابخانه های لازم خارج از بدنه ی بستار (closure)

یک روش بهینه که استفاده از آن بسیار توصیه می شود، تعریف ورژن نیازمندی های کتابخانه ی خود (library dependencies) خارج از بدنه ی بستار (closure) به غایت آسان تر می شود. بدین وسیله نگهداری و maintenance به غایت آسان تر می شود.

```
ext {
    // App dependencies
    junitVersion = '4.12'
    mockitoVersion = '1.10.19'
    powerMockito = '1.6.2'
    hamcrestVersion = '1.3'
}
dependencies {
    // Dependencies for local unit tests
    testCompile "junit:junit:$junitVersion"
    testCompile "org.mockito:mockito-all:$mockitoVersion"
    testCompile "org.hamcrest:hamcrest-all:$hamcrestVersion"
    testCompile "org.powermock:powermock-module-junit4:$powerMockito"
```

```
        testCompile "org.powermock:powermock-api-mockito:$ext.powerMockito"
    }
```

**توجه:** پس از قرار دادن بستار ext داخل فایل build اصلی (root)، می توانید، برای مثال، با پارامتر 'rootProject.ext.junitVersion\$' به راحتی به property ها و خواص آن دسترسی داشته باشید.

## build-2-8 و کامپایل ورژن های (flavor) مختلف از اپلیکیشن اندرویدی خود

### 2-8-1 انواع (build flavor) build

به طور پیش فرض، اندروید دو نوع build دارد: debug (ورژن debug دربردارنده کد برای اشکال زدایی و مقداری گزارش یا logging) و release (ورژن آماده و بهینه سازی شده برای اجرا). برای این انواع build،

می توانید flavor های مختلف در Gradle build ایجاد نمایید (build دو ورژن دارد که هر دو یک نسخه از اپلیکیشن هستند، اما flavor برای سرویس گیرنده و مشتری های مختلف تولید می شوند مانند نسخه ی پولی و رایگان از یک اپلیکیشن).

سیستم Gradle build قادر است flavor های مختلفی از یک اپلیکیشن تولید و مدیریت کند. product flavor بیانگر یک ورژن اختصاصی از اپلیکیشن می باشد. این امر امکانی را فراهم می کند تا بخش هایی از codebase یا منابع مورد استفاده ی پروژه برای نسخه های مختلف اپلیکیشن مورد نظر متفاوت باشد.

برای مثال، می توانید برای انواع دستگاه ها همچون گوشی یا تبلت (device category)، ورژن های مختلف (build varaint) تولید نمایید. به عنوان یک مورد استفاده ی دیگر می توان به نسخه ی پولی و رایگان اپلیکیشن مورد نظر اشاره کرد. همچنین زمانی که قصد دارید به هنگام اجرای تست بر روی اپلیکیشن، منابع و کلاس های متفاوت را بکار ببرید.

## 2-2-8- تعریف انواع flavor در فایل build

می توانید با استفاده از بستار productFlavors در محتوای فایل app/build.gradle، نسخه های مختلف از اپلیکیشن نهایی خود اعلان نمایید.

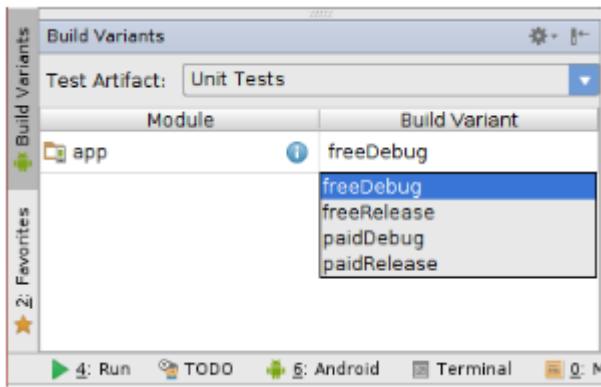
```
productFlavors {
    prod {
        applicationId = "com.vogella.android.gradlebuildflavors.prod"
        versionName = "1.0-paid"
    }
    mock {
        applicationId = "com.vogella.android.gradlebuildflavors.mock"
        versionName = "1.0-free"
    }
}
```

محتوای کل فایل build.gradle ممکن است ظاهری مشابه زیر داشته باشد.

```
apply plugin: 'com.android.application'
android {
    compileSdkVersion 22
    buildToolsVersion "22.0.1"
    defaultConfig {
        applicationId "com.exam.gradleexamples"
        minSdkVersion 19
        targetSdkVersion 22
        versionCode 1
        versionName "1.0"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
    productFlavors {
        prod {
            applicationId = "com.vogella.android.gradlebuildflavors.prod"
            versionName = "1.0-paid"
        }
        mock {
            applicationId = "com.vogella.android.gradlebuildflavors.mock"
            versionName = "1.0-free"
        }
    }
}
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.android.support:appcompat-v7:22.0.0'
```

```
testCompile 'junit:junit:4.+'
}
```

پس از اعلان flavor و ورژن های مختلف از اپلیکیشن، می توانید داخل محیط کاری Android Studio، تمامی ورژن های مختلف از اپلیکیشن خود را در کادر Build Variants مشاهده و انتخاب نمایید.



### ۸-۲-۳- ارائه‌ی منابع مختلف برای flavor ها و ورژن های مختلف اپلیکیشن

به منظور تعریف قابلیت و امکانات مختلف برای flavor مورد نظر، می بایست پوشه های مختلفی برای flavor های مشخص شده در پوشه‌ی app/src/ ایجاد نمایید.

منابع مختص به هر flavor، منابع کلی و اصلی را بازنویسی می کند. به طور مثال، زمانی یک آیکون متفاوت برای flavor معین تعیین کرده باشد، سیستم build و کامپایل Android، آن آیکونی که ویژه‌ی flavor مورد نظر تعریف شده را به عنوان آیکون اپلیکیشن انتخاب می کند.

### ۸-۴- تعریف source set های متفاوت برای flavor های مختلف از اپلیکیشن

دایرکتوری های مقیم در پوشه‌ی src در اصطلاح source sets شناخته می شود. هر flavor از نرم افزار می تواند source set اختصاصی خود را داشته باشد.

لازم به ذکر است که code file ها (فایل های حاوی کدهای جاوا) مانند resource ها و منابع پروژه جایگزین نمی شوند، بلکه با یکدیگر ادغام می گردند. به طور مثال، شما نمی توانید در یک az اپلیکیشن خود اکتیویتی com.example.MainActivity را در پوشه‌ی app/main/java/ داشته باشید.

و در flavor دیگر، نسخه‌ی پیاده سازی شده‌ی دیگری از activity مجبور داشته باشید. در صورتی که دو نسخه‌ی پیاده سازی شده از activity داشته باشید، یک پیغام خطا در خصوص وجود کلاس‌های تکراری (duplicate class) صادر می‌شود.

با این وجود می‌توانید با جلوگیری از ایجاد کلاس در source folder اصلی پروژه و سپس ایجاد یک کلاس مجزا در هر flavor، برای هر flavor از اپلیکیشن خود نسخه‌ی پیاده سازی شده‌ی متفاوتی از یک کلاس (برای مثال activity) داشته باشید.

## تمرین: ساخت اپلیکیشن‌های اندرویدی با flavor های مختلف

### هدف اصلی از این تمرین

در تمرین جاری، یک اپلیکیشن اندرویدی با دو flavor مختلف به ترتیب به نام‌های prod و mock ایجاد می‌کنید.

نسخه‌ی mock منابع متفاوتی از نسخه‌ی prod مورد استفاده قرار می‌دهد. در نمونه‌ی اول، فایل build strings.xml مستقر در folder/project بازنویسی (override) می‌شود. اینکه کدام ورژن باشد، از طریق کادر Build Variants قابل تعریف است.

### ساخت پروژه‌ی جدید اندروید

یک پروژه‌ی جدید بر اساس قالب آماده (template) Empty Activity و با اسم پکیج com.vogella.android.gradlebuildflavors ایجاد نمایید.

دو flavor جدید در فایل app/build.gradle به نام‌های "prod" و "mock" تعریف کنید.

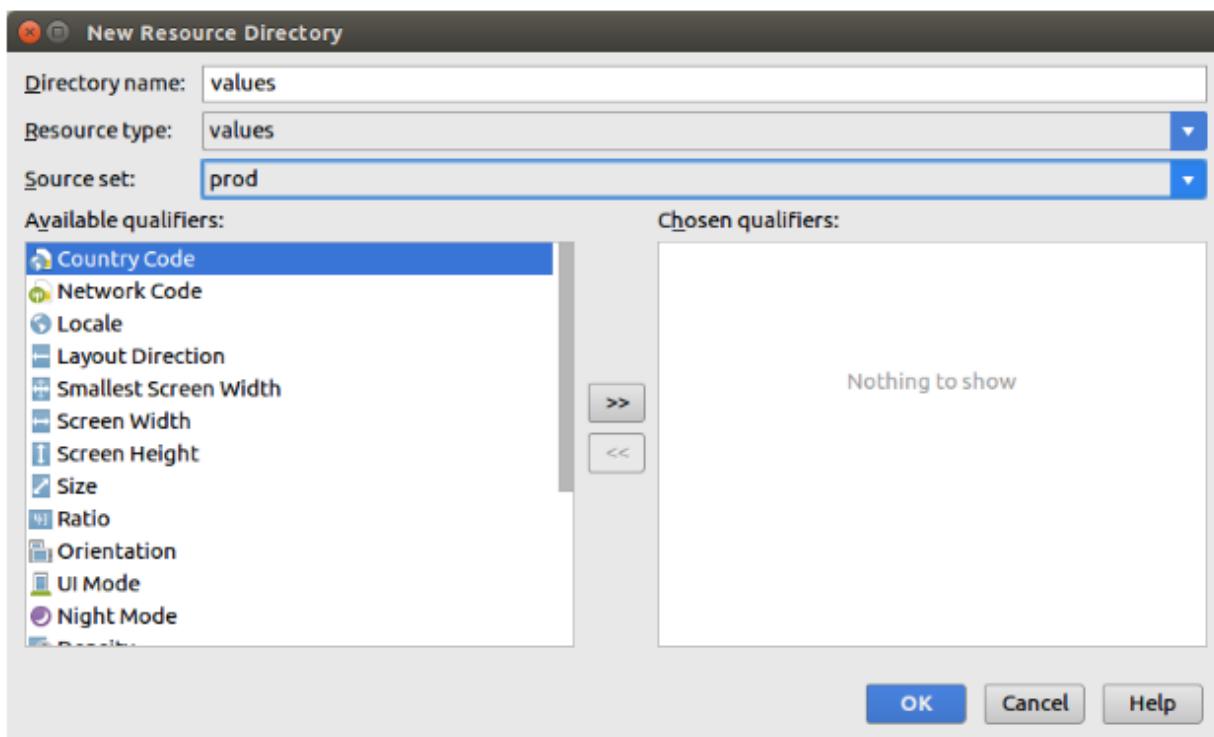
```
apply plugin: 'com.android.application'
android {
    compileSdkVersion 22
    buildToolsVersion "22.0.1"
    defaultConfig {
        applicationId "com.exam.gradleexamples"
```

```

minSdkVersion 19
targetSdkVersion 22
versionCode 1
versionName "1.0"
}
buildTypes {
    release {
        minifyEnabled false
        proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
    }
}
productFlavors {
    prod {
        applicationId = "com.vogella.android.gradlebuildflavors.prod"
        versionName = "1.0-paid"
    }
    mock {
        applicationId = "com.vogella.android.gradlebuildflavors.mock"
        versionName = "1.0-free"
    }
}
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.android.support:appcompat-v7:22.0.0'
    testCompile 'junit:junit:4.+'
}

```

ساختار درختی پوشه ها (folder structure) برای دو flavor (name) برده را ایجاد نمایید. اگر Android resource directory را انتخاب کنید، در آن صورت می توانید flavor مورد نظر را در پنجره‌ی محاوره ای زیر، همانند تصویر حاضر، انتخاب نمایید.

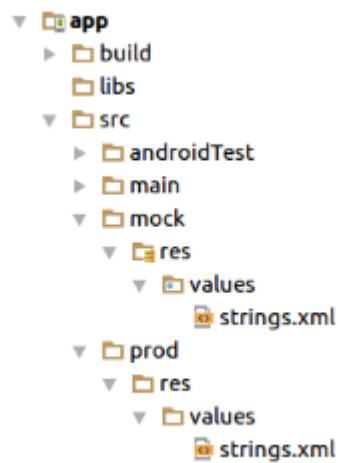


فایل strings.xml را از پوشه‌ی main به پوشه‌ی مناسب از flavor مربوطه کپی کرده و جایگذاری نمایید.

مقدار متغیر رشته‌ای strings.xml از فایل hello\_world را برای flavor به ترتیب به مقدارهای Prod World! و Mock World! تغییر دهید.

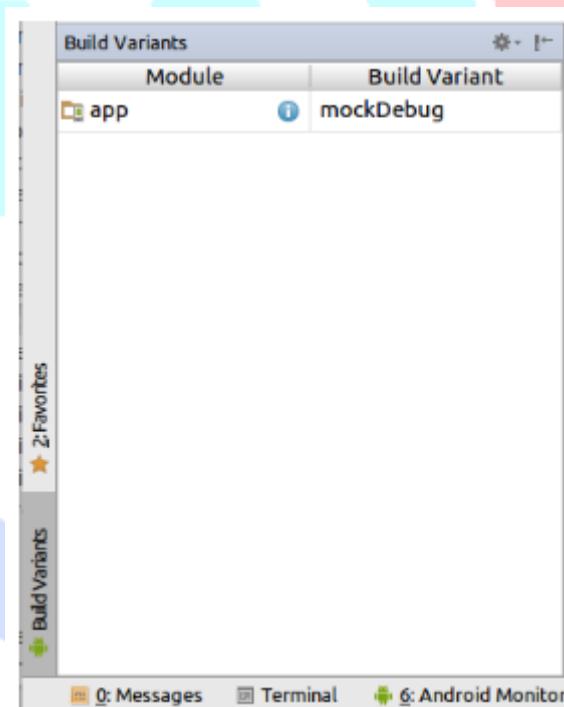
```
<resources>
    <string name="app_name">Flavor</string>
    <string name="hello_world">Mock World! </string>
</resources>

<resources>
    <string name="app_name">Flavor</string>
    <string name="hello_world">Prod World! </string>
</resources>
```



## ۲-۸-اعتبارسنجی و تست پروژه

در کادر Build Variant Android Studio محیط کاری گزینه `mockDebug` را انتخاب کرده و اپلیکیشن را جهت تست اجرا نمایید.



زمانی که اپلیکیشن را اجرا می کنید، می بایست رشته `Mock World!` را از `mock flavor` مشاهده نمایید. اکنون `prod flavor` را انتخاب کرده و اجرا نمایید. مقدار رشته ای مربوطه (`Prod Word!`) بایستی در نمایشگر قابل مشاهده باشد.

## 6-2-8-کامپایل و build پروژه از طریق خط دستور Gradle

می توانید با درج دستور `gradlew build`. در پنجره `cmd` فرمان، تمامی flavor های اپلیکیشن خود را اجرا نمایید.

## 7-2-8- تست ورژن یا flavor های مختلف از یک اپلیکیشن (gradle flavor های یک اپلیکیشن)

کلاسی به نام `ShareIntentBuilder` با پیاده سازی زیر ایجاد کنید. این کلاس به وسیله ی یک آبجکت `intent` که داده ها را به اشتراک می گذارد (`share intent`) و یک متده است `startSendActivity` دیگری را راه اندازی می نماید.

```
import android.content.Context;
import android.content.Intent;
public class ShareIntentBuilder {
    public static void startSendActivity(Context context, String title, String body) {
        Intent intent = new Intent();
        intent.setAction(Intent.ACTION_SEND);
        intent.putExtra(Intent.EXTRA_TITLE, title);
        intent.putExtra(Intent.EXTRA_TEXT, body);
        intent.setType("text/plain");
        Intent chooserIntent = Intent.createChooser(intent, context.getResources().getText(R.string.share));
        chooserIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        context.startActivity(chooserIntent);
    }
}
```

## 8-2-8-پیاده سازی نسخه های مختلف از کلاس MainActivity در flavor مورد نظر از اپلیکیشن

اجازه دهید `activity` ای که این آبجکت `intent` را در سطح خود تعریف کرده و به عنوان پارامتر به بیرون ارسال می کند، در `mock flavor` جایگزین گردد. چنانچه `mock` flavor "انتخاب شده بود، یک `activity` دیگر راه اندازی کنید که داده های ارسالی را نمایش می دهد. در صورتی که "prod" flavor انتخاب شده باشد، `intent` مشترک بین دو `activity` را ارسال کنید.

**توجه:** لازم به ذکر است که کلاس ها قابل بازنویسی نیستند. شما بایستی کلاس ها را در flavor های ویژه ای هر یک و نه در flavor اصلی ایجاد کنید.

### 3-8- تنظیم اختصاصی فایل Gradle build

#### 3-8-1- ویرایش اسم فایل apk خروجی

```
apply plugin: 'com.android.application'
android {
    // more
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
        applicationVariants.all { variant ->
            variant.outputs.each { output ->
                def file = output.outputFile
                def filename = file.name.replace("app", "lars")
                output.outputFile = new File(file.parent, filename)
            }
        }
    }
    // more
}
```

#### 3-8-2- تعریف keystore مجزا برای debug build

می توانید در فایل build.gradle خود یک keystore مجزا تعریف نمایید (برای درج امضای خالق یک پروژه ای اندرویدی بر روی فایل APK خروجی از keystore استفاده می شود). برای مشاهده ای جزئیات بیشتر به آدرس <http://tools.android.com/tech-docs/new-build-system/user-guide> مراجعه فرمایید.

به طور مثال می توانید Keystore را ویژه ای نسخه ای debug اپلیکیشن مورد نظر به صورت زیر ویرایش نمایید.

```
android {
    signingConfigs {
        debug {
            storeFile file("your.keystore")
        }
    }
}
```

}

## 4-8-انتقال / migrate کردن یک پروژه‌ی خروجی گرفته شده از Gradle به Eclipse

### 4-1-وارد کردن (import) یک پروژه‌ی خروجی گرفته شده از Eclipse در محیط کاری Android Studio

پروژه‌های اندرویدی به دو روش متفاوت (با دو نحوه‌ی تنظیم و پیکربندی) سازمان دهی می‌شوند: 1. یک روش پیکربندی قدیمی که تا سال 2013 توسط ابزار ساخت و توسعه‌ی اپلیکیشن‌های اندرویدی محیط کاری Eclipse ADT استفاده می‌شد. 2. دیگری از ساختار درختی جدید برای سازماندهی پروژه‌هی به نام Gradle build structure استفاده می‌کند. می‌توان سیستم Gradle را طوری تنظیم کرد که از هر دو فرمت ذکر شده برای پیکربندی پروژه استفاده کند.

پس از اینکه فایل Gradle مورد نیاز را به پروژه‌ی اندرویدی خروجی گرفته شده از Eclipse اضافه نمودید، می‌توانید پروژه‌ی خود را وارد محیط برنامه نویسی Android Studio کنید. برای این منظور کافی است داخل محیط مزبور، بر روی File از منوی اصلی کلیک کرده و گزینه‌ی Import Project را انتخاب نمایید. سپس پوشه‌ی پروژه و فایل Gradle build مربوطه را انتخاب کنید.

### 4-2-اضافه کردن فایل Gradle به پروژه‌ی اندرویدی که از محیط Eclipse خروجی گرفته شده است

برای نیل به این هدف، فایل build.gradle خود را به پوشه‌ی اصلی (root) پروژه اضافه نمایید.

```
buildscript {
    repositories {
        mavenCentral()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:2.2.0-beta3'
    }
}
apply plugin: 'com.android.application'
android {
    lintOptions {
        abortOnError false
    }
}
```

```

compileSdkVersion 22
buildToolsVersion "21.1.2"
    defaultConfig {
        targetSdkVersion 22
    }
sourceSets {
    main {
        manifest.srcFile 'AndroidManifest.xml'
        java.srcDirs = ['src']
        resources.srcDirs = ['src']
        aidl.srcDirs = ['src']
        renderscript.srcDirs = ['src']
        res.srcDirs = ['res']
        assets.srcDirs = ['assets']
    }
    // Move the build types to build-types/<type>
    // For instance, build-types/debug/java, build-types/debug/AndroidManifest.xml, ...
    // This moves them out of them default location under src/<type>/... which would
    // conflict with src/ being used by the main source set.
    // Adding new build types or product flavors should be accompanied
    // by a similar customization.
    debug.setRoot('build-types/debug')
    release.setRoot('build-types/release')
}
}

```

## بخش دوم :

# استفاده از سرور کامپایل و اجرای پروژه (build server) و Jenkins جهت build و اجرای اپلیکیشن های اندرویدی

آموزش حاضر اطلاعاتی در خصوص نحوه استفاده از سرور کامپایل پروژه‌ی Jenkins جهت build پروژه‌های اندرویدی در اختیار شما قرار می‌دهد.

## 5-8- کامپایل و اجرای پروژه های اندرویدی با Jenkins

### 5-8-1- پیش نیازهای کامپایل پروژه (build job) با Jenkins

Jenkins قادر است اپلیکیشن های اندرویدی را به صورت خودکار کامپایل و تست نماید. جهت انجام پروسه‌ی کامپایل و اجرای پروژه و ایجاد یک build job بر روی Jenkins، بایستی تنظیمات و ابزار مورد نیاز build را به صورت آماده داشته باشید. ابزار ساخت و توسعه‌ی اپلیکیشن اندروید خود به صورت پیش فرض یک فایل Gradle build معتبر تولید می‌کند که برنامه نویس می‌تواند از در Jenkins استفاده کند.

### 5-8-2- نصب مجموعه ابزار ساخت و توسعه‌ی نرم افزار اندروید (SDK)

لازم است Android SDK از قبل بر روی سرویس دهنده نصب باشد. در صورتی که سرور کامپایل و اجرای پروژه یا build server برنامه‌ی display server یا هماهنگ‌سازی ورودی و خروجی کاربر با باقی سیستم عامل، سخت افزار را نداشته باشد، بایستی Android SDK را از طریق خط دستور (command line) اجرا نمایید. همچنین لازم است مکان قرارگیری Android SDK جهت استفاده‌ی Jenkins را مشخص نمایید.

```
# download the Android SDK via wget
# for the correct URL see http://developer.android.com/sdk/index.html
# and look in the section "SDK Tools Only"
wget <link from the above website>
# example
# wget http://dl.google.com/android/android-sdk_r24.2-linux.tgz
# unpack it
tar zxvf filename
# Add the new directory to your path
# assumes you exported it to /opt/
# this must be set for the Jenkins user
export ANDROID_HOME="/opt/android-sdk-linux"
export PATH="$ANDROID_HOME/tools:$ANDROID_HOME/platform-tools:$PATH"
// run this as Jenkins user
```

```
android update sdk --no-ui
```

با استفاده از دستور android list targets می توانید ABI های نصب شده را مشاهده کنید یک رابط سطح-پایین بین اپلیکیشن و سیستم عامل یا یک (Application Binary Interface) نرم افزار دیگر ایجاد می نماید). در صورت عدم وجود ABI، لازم است آن را مانند زیر نصب نمایید.

```
android update sdk --all --filter sys-img-armeabi-v7a-android-23 --no-ui --force
```

### 3-5-8-نصب افزونه ها یا plug-in های Jenkins

جهت کامپایل و اجرای پروژه های اندرویدی (build) با Jenkins، لازم است افزونه های زیر را نصب نمایید:

- افزونه ی Gradle
- افزونه ی Git – در صورتی که Git برای منابع پروژه مورد استفاده قرار گرفته شده باشد.
- افزونه ی شبیه ساز اندروید (Android Emulator) – به توسعه دهنده امکان می دهد تا شبیه ساز اندروید (android emulator) را اجرا و فعال کرده و تا زمانی که شبیه ساز کاملا اجرا نشده، پروسه ی کامپایل و اجرای پروژه (build) را متوقف سازد. برای تست اپلیکشن های اندروید مورد نیاز می باشد.

### 4-5-8-ایجاد build job برای اپلیکیشن های اندرویدی

به منظور ایجاد build job های اندرویدی در Jenkins، پس از انتخاب New Job، اسم job مربوطه را وارد کنید. حال گزینه ی Build a free-style software project را انتخاب کنید.

Item name

**Freestyle project**  
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

**Maven project**  
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

**External Job**  
This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use Jenkins as a dashboard of your existing automation system. See [the documentation for more details](#).

**Multi-configuration project**  
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

**OK**

سپس می بایست مشخص کنید که منبع و source باید از کجا کپی (clone) شود.

**Source Code Management**

None  
 CVS  
 CVS Projectset  
 Git

Repositories      Repository URL

Credentials

Branches to build      Branch Specifier (blank for 'any')

دلخواه را انتخاب کنید. در تصویر بالا, job و عملیات زمان بندی شده مورد نظر، هر 15 دقیقه یکبار repository انتخابی (Git) را چک می کند.

**Build Triggers**

- Build after other projects are built
- Build periodically
- Poll SCM

Schedule

H/15 \* \* \* \*

Would last have run at Saturday, June

یک مرحله کامپایل و اجرا با سیستم Gradle build step (Gradle build step) Jenkins اضافه نمایید.

Install Android package  
Install Android project prerequisites  
Invoke Ant  
**Invoke Gradle script**  
Invoke top-level Maven targets  
Run Android monkey tester  
Uninstall Android package  
**Add build step ▾**

**Post-build Actions**

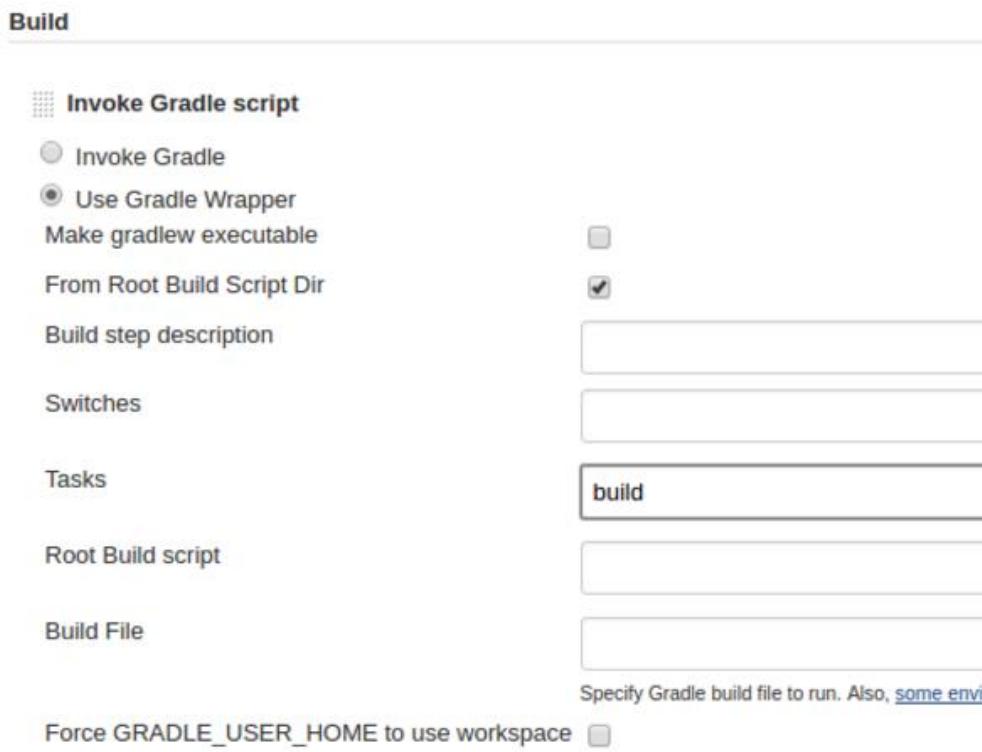
**Add post-build action ▾**

**Save** **Apply**

836

آدرس آموزشگاه : تهران - خیابان شریعتی - بالاتر از خیابان ملک - جنب بانک صادرات - پلاک 651 طبقه دوم - واحد 7

88146323 - 88446780 - 88146330



پس از طی نمودن مراحل فوق بایستی بتوانید Android build خود را اجرا نمایید.

### 5-5-8- اجرای تست بر روی دستگاه

جهت اجرای تست های instrumentation بر روی دستگاه دلخواه خود، باید یک دستگاه مجازی و شبیه ساز محیط (virtual device) اندروید راه اندازی نمایید.

دقت کنید که گزینه *Show emulator window* نبایستی انتخاب شده باشد چرا که عملیات زمان بندی شده یا job شما نباید به حضور یک برنامه *display server* وابسته باشد.

**Build Environment**

- Assign unique TCP ports to avoid collisions
- Run an Android emulator during build

Run existing emulator ?

Run emulator with properties ?

Android OS version

Screen density

Screen resolution

Device locale

SD card size

Target ABI

Emulator name suffix

Hardware

Add custom hardware property...

**Common emulator options**

- Reset emulator state at start-up ?
- Show emulator window ?
- Use emulator snapshots ?

**Advanced...**

جهت اجرای تست و تست های **instrumentation**، **bairsti** از **test** استفاده کنید. **connectedAndroidTest target**

**5-8- دیگر افزونه های مفید برای build پروژه های اندرویدی**

Build multi-configuration – از یک job دیگر با گزینه **Android Emulator Plugin** پشتیبانی می کند. این گزینه به شما امکان می دهد تا همزمان یک پروژه را بر

روی چندین نسخه‌ی تنظیم شده از شبیه ساز تست نمایید. به طور مثال می‌توانید پروژه را با زبان‌ها، چگالی پیکسلی و وضوح تصویر مختلف تست نمایید.

معمولًا دو job وجود دارد. یک job وظیفه‌ی build ساده‌ی یا کامپایل و اجرای تست بر روی پروژه را بر عهده دارد. دیگری یک job برای تست پروژه بر روی شبیه سازهایی با تنظیمات و پیکربندی متفاوت است.

افزونه‌ی پر کاربرد دیگری که بد نیست به آن اشاره شود، افزونه‌ی Lint است که به شما امکان می‌دهد تا ساختارهای مشکوک به داشتن خطا در کد پروژه را به واسطه‌ی Jenkins پیدا کرده و برطرف نمایید.

برای کسب اطلاعات بیشتر درباره‌ی این افزونه می‌توانید به آدرس <https://wiki.jenkins-ci.org/display/JENKINS/Android+Lint+Plugin> مراجعه نمایید.



## بخش سوم :

# استفاده از ابزار Android Debug Bridge (پل برقراری ارتباط و کنترل دستگاه یا شبیه ساز محیط اندرویدی و خطای دایی کد)

آموزش حاضر شرح می دهد چگونه با استفاده از ابزار ADB به دستگاه واقعی یا شبیه ساز محیط اندروید (AVD) متصل شده و آن را مدیریت کنید.

## 8-6-ابزار دسترسی، مدیریت و اشکال زدایی پروژه‌ی اندرویدی / ADB

ADB یا به اختصار Android Debug Bridge یک ابزار خط دستور جهت ارتباط و کنترل شبیه ساز (Emulator) یا دستگاه اندرویدی متصل به سیستم است و دستورات کاربردی متعددی نظیر نصب برنامه بر روی محیط شبیه ساز ، دسترسی به فایل های مستقر بر روی شبیه ساز و ... را برای برنامه نویس فراهم می کند.

می توانید با درج دستور adb در خط فرمان، به دستگاه اندرویدی خود دسترسی پیدا کنید. دسترسی و اتصال به یک دستگاه مجازی (AVD) به راحتی امکان پذیر می باشد، اما جهت دسترسی کامل به دستگاه واقعی اندروید لازم است که آن دستگاه root شود. هر چند که دستگاه های نشده نیز با adb قابل دسترسی هستند، اما در آن صورت توسعه دهنده صرفا در سطح یک کاربر عادی اجازه دسترسی خواهد داشت.

ابزار adb در پوشه‌ی [android-sdks]/platform-tools موجود می باشد. به منظور دسترسی کامل به این ابزار، بایستی آن را به متغیر path اضافه نمایید.

adb به شما امکان می دهد تا دستگاه اندرویدی خود ارسال کرده، فایل هایی را در آن قرار داده یا از آن واکنش نمایید. همچنین امکان دسترسی از خط فرمان به سرور راه دور (shell) و دستگاه اندروید را فراهم آورده و به توسعه دهنده اجازه می دهد تا اطلاعاتی نظیر میزان مصرف حافظه دستگاه را مشاهده نماید. مبحث زیر نحوه استفاده از دستورات مربوطه را برای شما شرح می دهد.

در صورتی که چندین دستگاه اندرویدی فعال و در حال اجرا داشته باشید، می توانید دستوراتی را به دستگاه ارسال کنید.

```
# Lists all devices
adb devices
#Result
List of devices attached
emulator-5554 attached
emulator-5555 attached
# Issue a command to a specific device
adb -s emulator-5554 shell
```

## 7-8-استفاده از adb

### 7-8-1-اجرای یک activity از خط دستور (command line)

#### کپی فایل ها

می توانید با درج دستور زیر در خط فرمان، یک فایل را از دستگاه کپی کرده یا در آن جایگذاری کنید.

```
adb shell am start -n yourpackagename/.activityname
```

## 8-8-حذف اپلیکیشن از دستگاه به وسیله ای دستورات adb

می توانید یک اپلیکیشن اندرویدی را با دستورات خط فرمان (از طریق shell) از دستگاه میزبان حذف نمایید. کافی است پوشه `data/app` را تغییر داده و سپس به راحتی اپلیکیشن اندرویدی را حذف نمایید.

در صورت تمایل می توانید یک اپلیکیشن را به وسیله `adb` دستورات `adb`, با ذکر اسم پکیج از دستگاه حذف نمایید.

```
adb uninstall <packagename>
```

## 9-اتصال به دستگاه با استفاده از Telnet

در صورت تمایل می توانید بجای استفاده از `adb`, به واسطه `telnet` به دستگاه اندرویدی دلخواه وصل شوید. این به شما امکان می دهد تا برخی از بسترهای همچون تماس دریافتی را شبیه سازی کنید، وضعیت اتصال به شبکه را ویرایش نموده و وضعیت جغرافیایی خود را تنظیم کنید. به منظور اتصال به دستگاه شبیه سازی شده کافی است از "telnet localhost 5554" استفاده کنید. جهت خروج از فضای کاری `console` نیز کافی است دستور `quit` یا `exit` را اجرا نمایید.

به طور مثال، جهت تغییر تنظیمات گوشی، یا به منظور دریافت `sms` و تماس دریافتی، می توانید از دستورات زیر استفاده کنید.

```
# connects to device
telnet localhost 5554
# set the power level
power status full
power status charging
# make a call to the device
gsm call 012041293123
# send a sms to the device
sms send 12345 Will be home soon
# set the geo location
geo fix 48 51
```

## 10-دریافت اطلاعات سیستمی با ابزار خط دستور dumpsys

## adb dumpsys 8-10-1

دستور فوق به شما اجازه می دهد تا اطلاعات مربوط به سیستم اندروید و اپلیکیشن های در حال اجرا ضبط کرده و آن ها را رصد نمایید.

جهت بازبایبی اطلاعات مربوطه به میزان مصرف حافظه، می توانید دستور زیر را فراخوانی کنید.  
**adb shell dumpsys meminfo <package.name>**

## dumpsys 8-10-2

دستور adb shell procrank تمامی اپلیکیشن های نصب شده بر روی دستگاه را بر اساس میزان مصرف حافظه لیست می کند. لازم به ذکر است که دستور نام بردۀ بر روی دستگاه واقعی اندروید کار نمی کند. جهت واکشی لیست اپلیکیشن ها بر اساس میزان مصرف حافظه در دستگاه واقعی، بایستی دستور adb shell dumpsys meminfo را فراخوانی کنید.

## 8-10-3 واکشی اطلاعات درباره ی عملیات و تسك های زمان بندی شده

جهت کسب اطلاعات در خصوص اپلیکیشن، کافی است دستور adb shell dumpsys alarm را همراه با اسم پکیج پروژه در خط فرمان فراخوانی کنید. خروجی ممکن است ظاهری مشابه زیر داشته باشد:

```
RTC #6: Alarm{434a1234 type 1 com.example}
  type=1 whenElapsed=608198149 when=+12m13s122ms window=-1 repeatInterval=0 count=0
  operation=PendingIntent{430cf612: PendingIntentRecord{*43bbf887* com.vogella startService}}
```

همان طور که مشاهده می کنید، هشدار برای حدودا 12 دقیقه تنظیم شده است.

جهت کسب اطلاعات لازم در خصوص intent pending intent معلق)، دستور adb shell dumpsys activity intents (در این مثال PendingIntentRecord را اجرا کرده و ID مربوط به PendingIntentRecord تلفیقی از درخواست عملیات (راه 43bbf887) را به عنوان پارامتر ارسال کنید (PendingIntent) از درخواست عملیات (راه broadcast، جزییات عملیات اندازی یک activity دیگر، اجرا و فراخوانی یک سرویس یا ارسال یک intent (context) است. نمونه ایجاد شده می تواند به سایر اپلیکیشن ها ارسال (در قالب intent) گردد تا با کمک آنها حتی پس از خاتمه یافتن اپلیکیشن تولید کننده ای pendingintent و حذف

آن از حافظه، عملیاتی معینی انجام گیرد. همچنین آبجکت PendingIntent با برخورداری از flag هایی که عملکرد آنرا با توجه به اینکه نمونه ای از آن کلاس pendingintent موجود است یا نه، مدیریت می نماید:

```
* PendingIntentRecord{43bbf887 com.vogella startService}
uid=10042 packageName=com.vogella type=startService flags=0x0
requestIntent=act=MY_ACTION cmp=com.vogella/.MyService (has extras)
```

## تسک هایی که اطلاعات مربوط به مصرف باتری را واکشی می کنند

از ویرایش 5.0 اندروید، شما می توانید با فراخوانی دستور زیر در خط فرمان، اطلاعات مربوط به میزان مصرف باتری را واکشی نمایید:

```
adb shell dumpsys batterystats --charged <package-name>
```





بخش اول :

845

آدرس آموزشگاه : تهران - خیابان شریعتی - بالاتر از خیابان ملک - جنب بانک صادرات - پلاک 651 طبقه دوم - واحد 7

88146323 - 88446780 - 88146330

# توابع کتابخانه ای مکان یابی / Location API اندروید و ابزار fused location provider

آموزش حاضر نحوه‌ی استفاده از ابزار fused location provider را به شما آموزش می‌دهد.

## دواخته داده‌ها در اندروید / Android Location API

بدست آوردن اطلاعات مربوط به مکان جغرافیایی دستگاه امروزه اغلب دستگاه‌های مبتنی بر سیستم عامل اندروید این امکان را برای کاربران فراهم می‌کنند تا مکان جغرافیایی جاری دستگاه را بدست بیاورند. این امر از طریق مازول GPS، wifi یا تکنیک مثلث سازی برج‌های مخابراتی امکان‌پذیر می‌باشد.

با استفاده از Google Play بازیابی مکان جغرافیایی دستگاه جاری را بازیابی می‌کند.

### نصب

به منظور استفاده از location manager، سرویس Google play را در قالب dependency به فایل build.gradle اضافه نمایید.

```
dependencies {
    compile 'com.google.android.gms:play-services:9.2.0'
    compile 'com.google.android.gms:play-services-location:9.2.0'
}
```

همچنین بایستی مجوز نیاز را در فایل تنظیمات اپلیکیشن اندرویدی، manifest، اعلان نمایید.

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

### استفاده از LocationManager

اکنون می‌توانید به آخرین مکان جغرافیایی دستگاه دسترسی داشته باشید. fuse location provider توابع کتابخانه‌ای نوین و کارایی در اختیار توسعه دهنده قرار می‌دهد که کار مکان

یابی را انجام می دهد. در زیر مثالی را مشاهده می کنید که از این توابع برای واکنش اطلاعات مکان جاری دستگاه استفاده می کند.

## بدست آوردن مختصات جغرافیایی به صورت دو طرفه (forward and reverse geocoding)

کلاس Geocoder به شما این امکان را می دهد تا مختصات جغرافیایی (عرض و طول جغرافیایی) یک مکان را بر اساس آدرس ارائه شده بدست آورده و بالعکس (بر اساس عرض و طول جغرافیایی ارائه شده، آدرس دقیق مثل اسم خیابان محل قرارگیری دستگاه را بدست آورید). از این پروسه تحت عنوان reverse geocoding یاد می شود. کلاس نام برده از یک سرویس آنلاین Google استفاده می کند.

### امنیت

اگر می خواهید به حسگر GPS دسترسی داشته باشید، در آن صورت لازم است مجوز ACCESS\_FINE\_LOCATION را نیز اعلان نمایید. در غیر این صورت باید ACCESS\_COARSE\_LOCATION را در فایل تنظیمات اعلان کنید.

## درخواست از کاربر برای فعال سازی حسگر GPS (امکان سخت افزاری GPS یا مکان یاب)

کاربر باید تصمیم بگیرد که امکان سخت افزاری GPS فعال شود یا خیر.

می توانید با فراخوانی متد isProviderEnabled() مطمئن شوید آیا LocationManager فعال و پیاده سازی شده است یا خیر (LocationManager کلاسی است که امکان دسترسی به سرویس های مکان یابی سیستم را فراهم می کند). در صورتی که فعال نشده باشد، می توانید به واسطه ارسال یک آبجکت Intent که مقدار خاصیت name از المان action آن (اسم action) در فایل xml بر روی Settings.ACTION\_LOCATION\_SOURCE\_SETTINGS تنظیم شده به کلاس android.provider.Settings.کاربر را به بخش تنظیمات مربوطه ارجاع دهید.

```
LocationManager service = (LocationManager) getSystemService(LOCATION_SERVICE);
boolean enabled = service
    .isProviderEnabled(LocationManager.GPS_PROVIDER);
```

```
// check if enabled and if not send user to the GSP settings
// Better solution would be to display a dialog and suggesting to
// go to the settings
if (!enabled) {
    Intent intent = new Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS);
    startActivity(intent);
}
```

معمولاً به این صورت است که توسعه دهنده یک `AlarmDialog` در نمایشگر باز می کند که از کاربر می پرسد آیا می خواهد امکان سخت افزاری GPS را فعال کند یا اینکه اجرای اپلیکیشن مربوطه باید کلا لغو گردد.

لازم به ذکر است که شما نمی توانید به طور مستقیم امکان GPS را در کد برنامه فعال کنید بلکه کاربر باید آن را به صورت دستی فعال کند.

## استفاده از امکان GPS و تنظیم موقعیت جاری

### فعال سازی امکان GPS در محیط شبیه ساز

لازم است امکان GPS را بر روی دستگاه تست گیری خود فعال نمایید. اگر اپلیکیشن را بر روی محیط شبیه ساز تست نمایید، در حالی که امکان GPS فعال نشده باشد، کلاس `LocationManager` مقدار `null` را بر می گرداند.

یا صفحه `i` `activity` یا صفحه `Google Map`، بایستی به صورت خودکار امکان سخت افزاری GPS را در شبیه ساز فعال کند. اما چنانچه می خواهید از `location manager` به طور مستقیم استفاده نمایید، بایستی این کار را خود انجام دهید. در حال حاضر مشکلاتی برای استفاده ای مستقیم از `location manager` وجود دارد.

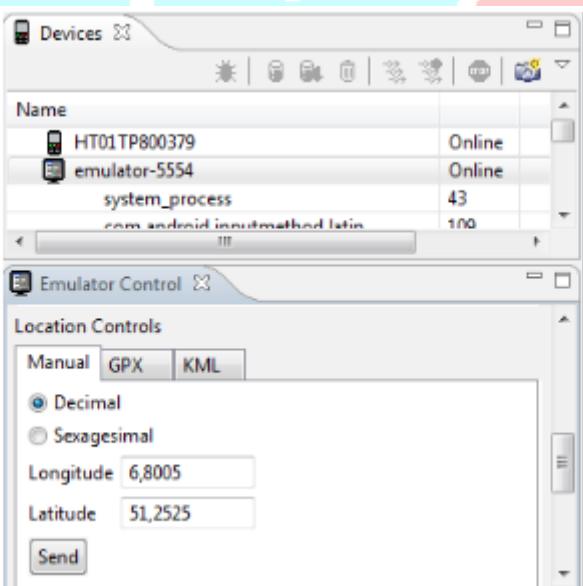
`Google Maps` را در محیط شبیه ساز اجرا کرده و درخواست اطلاعات موقعیت جغرافیایی جاری را بدھید. این کار به شما اجازه می دهد تا GPS را فعال نمایید. مختصات جدید GPS را به شبیه ساز اندروید ارسال نمایید.

## تنظیم موقعیت جغرافیایی

می توانید با استفاده از "DDMS" Perspective، موقعیت جغرافیایی خود را به شبیه ساز یا دستگاه متصل ارسال نمایید. برای این منظور، ابتدا Perspective را باز کرده و **.Window ▶ Open Perspective ▶ Other... ▶ DDMS** سپس مسیر رو به رو را طی نمایید: (perspective) اسمی که به مجموعه ای از view ها و ناحیه‌ی ویرایشگر اشاره می‌کاری اشاره دارد) Eclipse

که سرnam واژگان DDMS است کاراترین ابزار اشکال زدایی برای پروژه‌های اندرویدی بوده و از قابلیت‌های آن می‌توان به گزارش‌گیری و نمایش اطلاعات file manager اشاره کرد. این ابزار قادر است به عنوان یک برنامه‌ی جدا راه اندازی شده و بر دستگاه میزبان نظارت داشته باشد.)

می‌توانید در **Emulator Control**، مختصات جغرافیایی را وارد کرده و دکمه‌ی **Send** را فشار دهید.



همچنین می‌توانید مختصات جغرافیایی را در شبیه ساز Android از طریق telnet تنظیم نمایید. پنجره‌ی فرمان یا کنسول را باز کرده و به دستگاه دلخواه متصل شوید. شماره‌ی port دستگاه شما در نوار عنوان (title area) شبیه ساز قابل مشاهده می‌باشد.

telnet localhost 5554

حال از طریق دستور زیر موقعیت جغرافیایی را مقداردهی و تنظیم نمایید.  
geo fix 13.24 52.31

## آموزش: استفاده از توابع کتابخانه ای Location API اندروید

### ساخت پروژه

یک پروژه **دیجیتال** و **de.vogella.android.locationapi.simple** جدید به ترتیب به نام **ShowLocationActivity** ایجاد نمایید.

گفتنی است که این مثال از Google Map استفاده نمی کند و بجای محیط شبیه ساز بر روی دستگاه واقعی اندروید اجرا می شود.

فایل layout خود را از `<filename class="directory">res/layout` به کد زیر ویرایش نمایید.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <LinearLayout
        android:id="@+id/linearLayout1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="40dip"
        android:orientation="horizontal" >
        <TextView
            android:id="@+id/TextView01"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="10dip"
            android:layout_marginRight="5dip"
            android:text="Latitude: "
            android:textSize="20dip" >
        </TextView>
        <TextView
            android:id="@+id/TextView02"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
```

850

آدرس آموزشگاه: تهران - خیابان شریعتی - بالاتر از خیابان ملک - جنب بانک صادرات - پلاک 651 طبقه دوم - واحد 7

88146323 - 88446780 - 88146330

```

        android:text="unknown"
        android:textSize="20dip" >
    </TextView>
</LinearLayout>
<LinearLayout
    android:id="@+id/linearLayout2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >
    <TextView
        android:id="@+id/TextView03"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dip"
        android:layout_marginRight="5dip"
        android:text="Longitude: "
        android:textSize="20dip" >
    </TextView>
    <TextView
        android:id="@+id/TextView04"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="unknown"
        android:textSize="20dip" >
    </TextView>
</LinearLayout>
</LinearLayout>

```

## افزودن مجوزهای لازم

مجوزهای زیر را در فایل تنظیمات اپلیکیشن (AndroidManifest.xml) اعلان نمایید.

- INTERNET
- ACCESS\_FINE\_LOCATION
- ACCESS\_COARSE\_LOCATION

## ویرایش کلاس Activity

محتوای کلاس ShowLocationActivity را به صورت زیر ویرایش نمایید. با اعمال تغییرات زیر از کوئری گرفته شده و مقادیر مورد درخواست (خروجی) در activity manger گذاشته می شوند.

```

package de.vogella.android.locationsapi.simple;
import android.app.Activity;
import android.content.Context;
import android.location.Criteria;
import android.location.Location;

```

```

import android.location.LocationListener;
import android.location.LocationManager;
import android.os.Bundle;
import android.widget.TextView;
import android.widget.Toast;
public class ShowLocationActivity extends Activity implements LocationListener {
    private TextView latituteField;
    private TextView longitudeField;
    private LocationManager locationManager;
    private String provider;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        latituteField = (TextView) findViewById(R.id.TextView02);
        longitudeField = (TextView) findViewById(R.id.TextView04);
        // Get the location manager
        locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
        // Define the criteria how to select the locatioin provider -> use
        // default
        Criteria criteria = new Criteria();
        provider = locationManager.getBestProvider(criteria, false);
        Location location = locationManager.getLastKnownLocation(provider);
        // Initialize the location fields
        if (location != null) {
            System.out.println("Provider " + provider + " has been selected.");
            onLocationChanged(location);
        } else {
            latitudeField.setText("Location not available");
            longitudeField.setText("Location not available");
        }
    }
    /* Request updates at startup */
    @Override
    protected void onResume() {
        super.onResume();
        locationManager.requestLocationUpdates(provider, 400, 1, this);
    }
    /* Remove the locationlistener updates when Activity is paused */
    @Override
    protected void onPause() {
        super.onPause();
        locationManager.removeUpdates(this);
    }
    @Override
    public void onLocationChanged(Location location) {
        int lat = (int) (location.getLatitude());
        int lng = (int) (location.getLongitude());
        latitudeField.setText(String.valueOf(lat));
        longitudeField.setText(String.valueOf(lng));
    }
}

```

```

@Override
public void onStatusChanged(String provider, int status, Bundle extras) {
    // TODO Auto-generated method stub
}
@Override
public void onProviderEnabled(String provider) {
    Toast.makeText(this, "Enabled new provider " + provider,
        Toast.LENGTH_SHORT).show();
}
@Override
public void onProviderDisabled(String provider) {
    Toast.makeText(this, "Disabled provider " + provider,
        Toast.LENGTH_SHORT).show();
}
}

```

## اجرا و تست

اگر از شبیه ساز استفاده می کنید، مختصات جغرافیایی مورد نظر را به دستگاه ارسال کنید. با کلیک بر روی دکمه، مختصات جغرافیایی به نمایش گذاشته می شود.

# DADEHA

**بخش دوم :**

## توابع کتابخانه ای Google Maps / Google Maps v2

**Android API v2**

آموزش حاضر به شرح توابع کتابخانه ای Google Maps و نحوه‌ی استفاده از آن‌ها در پروژه‌های اندرویدی می‌پردازد. پروژه‌این مبحث در ویرایش 4.4 محیط کاری Eclipse، بر پایه‌ی ورژن 1.7 زبان جاوا و 4.4 اندروید نوشته شده‌است.

**Google Maps**

**MapView**

شرکت گوگل یک کتابخانه از طریق سرویس Google play در اختیار توسعه دهنده قرار می‌دهد که امکان استفاده‌ی آسان از Google Maps و پیاده سازی توابع آن در پروژه‌ی اندرویدی را به راحتی فراهم می‌آورد. مبحث حاضر از توابع کتابخانه ای Google Maps v2 استفاده می‌کند که نسبت به ورژن قبلی آن از قابلیت‌های بیشتری برخوردار است.

کتابخانه‌ی مذبور دو کلاس به ترتیب به نام `MapView` و `com.google.android.gms.maps.MapFragment` ارائه می‌دهد که کامپوننت نقشه را برای کاربر نمایش می‌دهد.

برای استفاده از Google Maps لازم است اطلاعات اضافی به فایل تنظیمات اپلیکیشن  
اضافه نمایید. **AndroidManifest.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.vogella.android.maps">
    <!--
        The ACCESS_COARSE/FINE_LOCATION permissions are not required to use
        Google Maps Android API v2, but you must specify either coarse or fine
        location permissions for the 'MyLocation' functionality.
    -->
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <!--
            The API key for Google Maps-based APIs is defined as a string resource.
            (See the file "res/values/google_maps_api.xml").
            Note that the API key is linked to the encryption key used to sign the APK.
            You need a different API key for each encryption key, including the release key that is used to
            sign the APK for publishing.
            You can define the keys for the debug and release targets in src/debug/ and src/release/.
        -->
        <meta-data
            android:name="com.google.android.geo.API_KEY"
            android:value="@string/google_maps_key" />
        <activity
            android:name=".MapsActivity"
            android:label="@string/title_activity_maps">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

## کلاس MapFragment

کلاس MapFragment از کلاس Fragment ارث بری کرده و توابعی جهت مدیریت چرخهٔ حیات (life cycle) و سرویس‌های لازم برای نمایش ویجت (کامپوننت رابط کاربری) GoogleMap را فراهم می‌آورد. GoogleMap کلاسی است که وظیفهٔ نمایش نقشه را بر عهده دارد. MapFragment برای دسترسی به این کلاس متدهای getMap() را فراخوانی می‌کند.

کلاس LatLng پل ارتباطی جهت تعامل با کلاس GoogleView خواهد بود.

### درج (نشانه) marker بر روی نقشه

می توانید با استفاده از کلاس Marker، بر روی نقشه نشانه درج نمایید. این قابلیت بسیار منعطف بوده و قابل تنظیم می باشد.  
کد زیر مثالی عملی را به نمایش می گذارد.

```
public class MainActivity extends Activity {
    static final LatLng HAMBURG = new LatLng(53.558, 9.927);
    static final LatLng KIEL = new LatLng(53.551, 9.993);
    private GoogleMap map;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        map = ((MapFragment) getFragmentManager().findFragmentById(R.id.map))
            .getMap();
        if (map!=null){
            Marker hamburg = map.addMarker(new MarkerOptions().position(HAMBURG)
                .title("Hamburg"));
            Marker kiel = map.addMarker(new MarkerOptions()
                .position(KIEL)
                .title("Kiel")
                .snippet("Kiel is cool")
                .icon(BitmapDescriptorFactory
                    .fromResource(R.drawable.ic_launcher)));
        }
    }
}
```

می توانید بر روی نمونه ی ساخته شده از روی کلاس GoogleMap، توابع listener برای گوش دادن به marker های درج شده بر روی نقشه الصاق نمایید. کلاس OnMarkerClickListener یک متده به نام onMarkerClicked(Marker) در اختیار توسعه دهنده قرار می دهد که به مجرد کلیک کاربر بر روی marker فراخوانی می شود.

### تنظیم اختصاصی GoogleMap

می توان GoogleMap را مطابق نیاز به صورت اختصاصی تنظیم کرد.  
کد زیر از وبسایت اصلی گوگل اقتباس شده است.

```

static final LatLng HAMBURG = new LatLng(53.558, 9.927);
static final LatLng KIEL = new LatLng(53.551, 9.993);
private GoogleMap map;
... // Obtain the map from a MapFragment or MapView.
//Move the camera instantly to hamburg with a zoom of 15.
map.moveCamera(CameraUpdateFactory.newLatLngZoom(HAMBURG, 15));
// Zoom in, animating the camera.
map.animateCamera(CameraUpdateFactory.zoomTo(10), 2000, null);

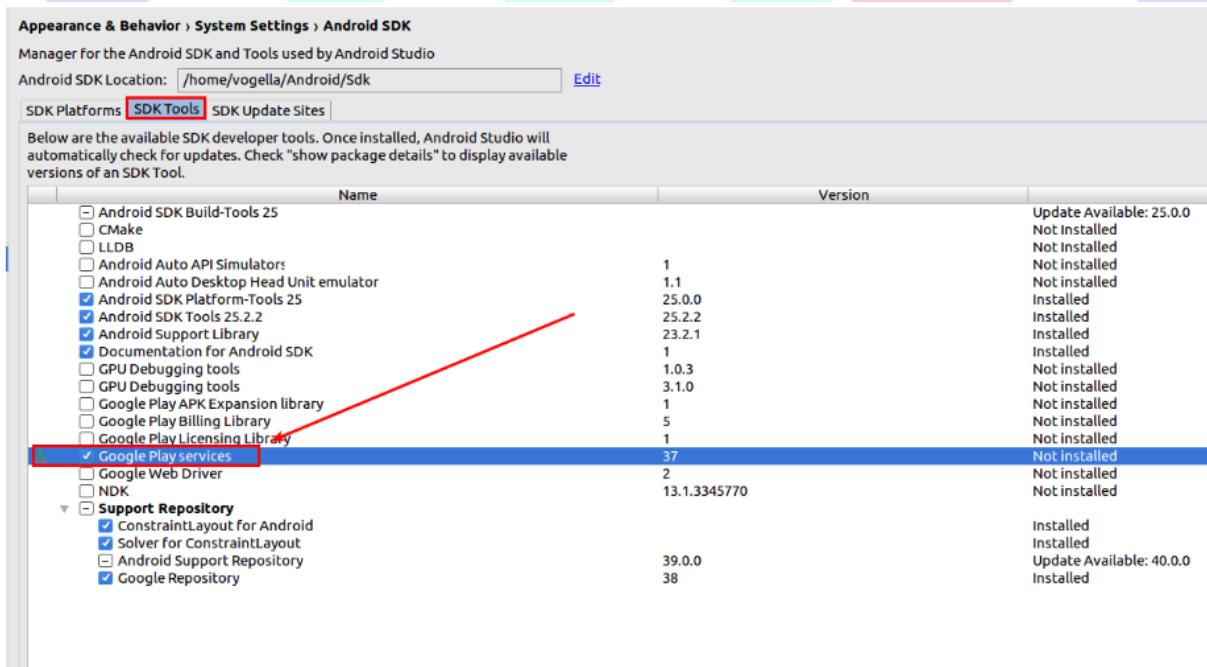
```

## نرم افزار شبیه ساز محیط اندروید (emulator) و Google Maps

لازم است یک شبیه ساز مبتنی بر توابع کتابخانه ای Google ایجاد نمایید. با استفاده از این شبیه ساز می توانید بعده ها Google map و دیگر سرویس های ادغام شدنی نظیر Play Service را تست نمایید.

## نصب Google Play service دانلود Google Play Service

Google Play Services را باز کرده و Android SDK Manager



اکنون می توانید dependency مورد نیاز را به فایل app/build.gradle اضافه نمایید.

```
compile 'com.google.android.gms:play-services:9.8.0'
```

## ساخت و دریافت کلید شناسگر (key) Google Map

جهت استفاده از Google Maps لازم است یک شناسه گر برنامه‌ی فراخواننده یا API key معتبر Google ایجاد نمایید. این کلید رایگان بوده و می‌توان از آن برای تمامی اپلیکیشن‌هایی که توابع کتابخانه‌ی Maps را فراخوانی می‌کنند، استفاده نمایید. در خصوص استفاده از این کلید محدودیتی وجود نداشته و قابلیت پشتیبانی از بی‌نهایت کاربر را دارد.

برای دریافت این کلید بایستی از طریق Google APIs Console اقدام نمایید. جهت دریافت کلید لازم است اسم پکیج و امضای دیجیتالی (signature key) اپلیکیشن را ارائه نمایید.

منظور از امضای دیجیتالی همان کلیدی است که اپلیکیشن اندروید خود را به هنگام خروجی گرفتن، توزیع و نصب (مرحله‌ی deployment) با آن امضا و به طور منحصر بفرد نشانه گذاری می‌کنید. در حین توسعه‌ی اپلیکیشن، سیستم کامپایل و اجرای پروژه‌های اندرویدی (build system)، به صورت خودکار یک debug key تولید کرده و از آن جهت اشکال زدایی اپلیکیشن و اینکه کاربر مجبور نباشد هر بار برای ترجمه و اجرای برنامه امضای الکترونیکی را وارد کند، استفاده می‌نماید.

### ایجاد SHA-1 برای امضای دیجیتالی (Signature key)

یا کلید اشکال زدایی و کامپایل پروژه تحت آدرس debug key userhome/.android/debug.keystore قابل دسترسی می‌باشد.

SHA-1 (الگوریتم درهم سازی اینمن): تابع درهم سازی در مبحث رمزنگاری است.

برای ساخت SHA-1 ویژه‌ی debug keystore برای اعتبارسنجی شناسه‌ی برنامه نویس و توسعه دهنده‌ی اپلیکیشن استفاده می‌شود.). می‌توانید دستور keytool را از JDK نصب شده اجرا کنید.

```
keytool -list -v -alias androiddebugkey \
-keystore <path_to_debug_keystore>debug.keystore \
-storepass android -keypass android
```

لازم است خروجی-1 SHA را برای استفاده در آینده کپی نمایید.

## ثبت نام در Google APIs Console

شما بایستی در Google APIs Console ثبت و اعلان نمایید که می خواهید از توابع Google Services برای پروژه‌ی اندرویدی خود استفاده نمایید. از منوی My Project المان Maps را انتخاب نمایید.



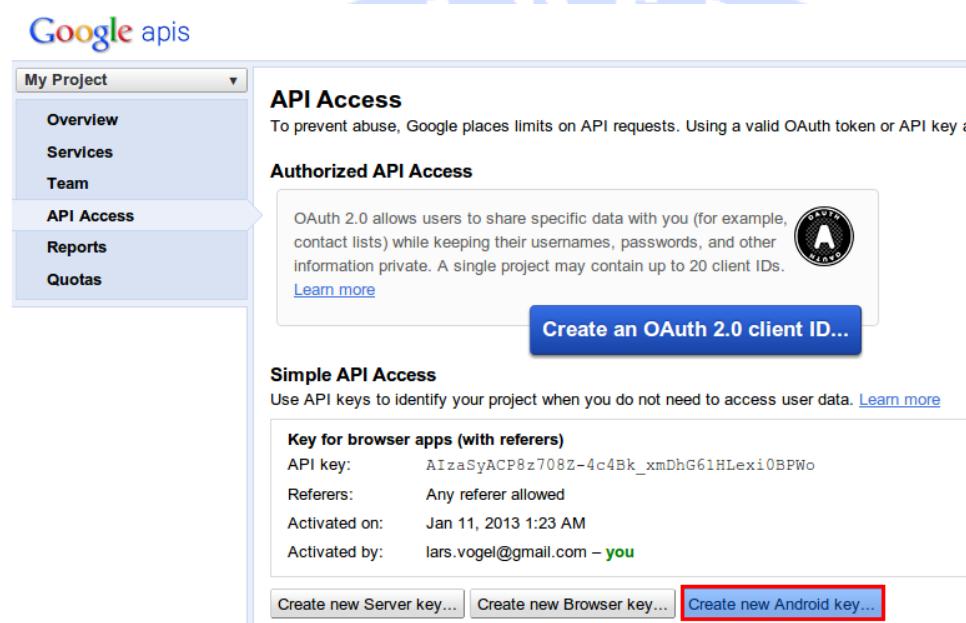
Google Maps Android API v2 را فعال نمایید.



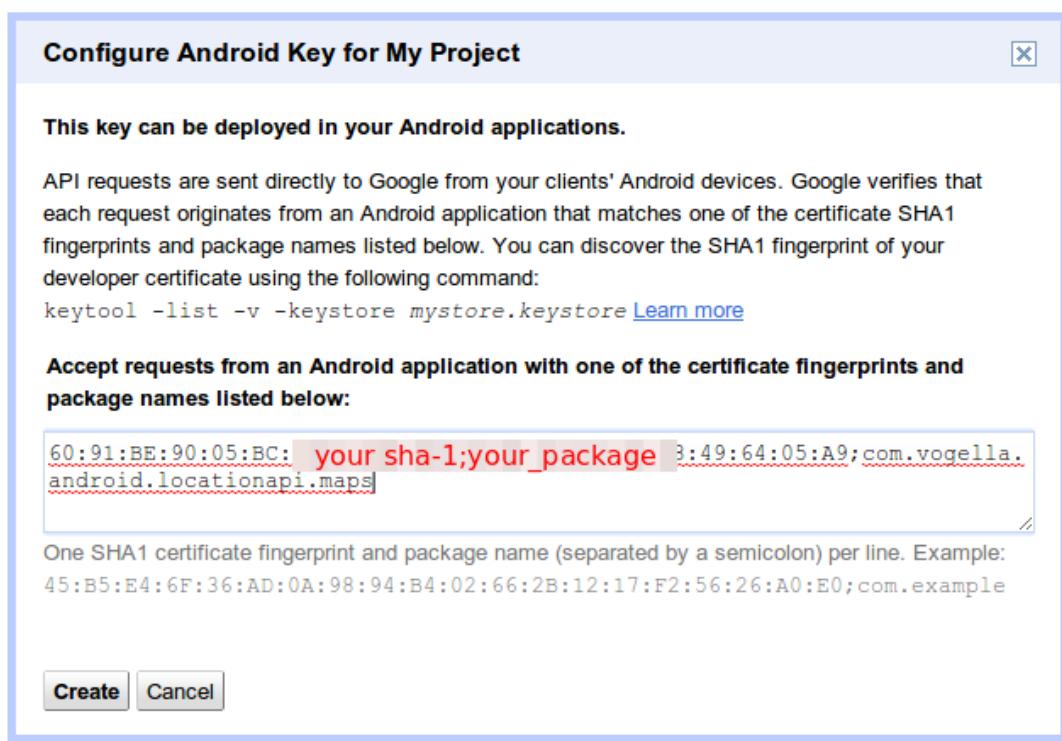
## ساخت کلید و امضای دیجیتالی برای اپلیکیشن

شما بایستی بعده ها اپلیکیشن را از طریق اسم پکیج در کنسول همراه با SHA-1 fingerprint ثبت نمایید (signature key)؛ دنباله‌ای کوتاه از بایت‌ها می‌باشد که برای تایید اعتبار و یا جستجوی کلید عمومی/public key طولانی‌تر استفاده می‌شود. اثرانگشت‌ها با اعمال یک تابع درهم‌ساز رمزنگارانه/hash function بر روی یک کلید عمومی ایجاد می‌گردند. از آنجا که اثر انگشت‌ها نسبت به کلیدهای مربوطه کوتاه‌تر هستند، می‌توان از آن‌ها به منظور ساده سازی برخی وظایف مدیریت کلید استفاده نمود. رمزنگاری کلید عمومی

یا رمزگاری نامتقارن روشی از رمزگاری است که کلید مورد استفاده برای رمزگذاری با کلید مربوط برای رمزگشایی با هم متفاوت است.). برای این منظور پس از کلیک بر روی المان Create new Android key... API Access را انتخاب نمایید. سپس بر روی المان... کلیک کنید.



حال در کادر مربوطه SHA-1 fingerprint و اسم پکیج اپلیکیشن خود را وارد کرده و آن ها را با ویرگول از هم جدا نمایید. برای مثال همان طور که در تصویر زیر مشاهده می کنید، پکیج com.vogella.android.locationapi.maps در کادر مربوطه درج شده است.



## آموزش: استفاده از Google Maps در پروژه

در این آموزش یک اپلیکیشن اندرویدی می نویسید که GoogleMap را از طریق یک fragment اضافه کند. برای کاربر نمایش می دهد.

### نصب Google Play Service

لازم است Google Play Service را نصب نمایید.

### ایجاد پروژه اندرویدی

یک پروژه اندرویدی به نام com.vogella.android.maps بر اساس قالب آماده ای Google Maps Activity (template) ایجاد نمایید.

### بررسی فایل تنظیمات پروژه

محتوای فایل `manifest` را چک کرده و مطمئن شوید که مجوزهای زیر توسط `template` نام بردۀ در آن اعلان شده اند.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.vogella.android.maps">
    <!--
        The ACCESS_COARSE/FINE_LOCATION permissions are not required to use
        Google Maps Android API v2, but you must specify either coarse or fine
        location permissions for the 'MyLocation' functionality.
    -->
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <!--
            The API key for Google Maps-based APIs is defined as a string resource.
            (See the file "res/values/google_maps_api.xml").
            Note that the API key is linked to the encryption key used to sign the APK.
            You need a different API key for each encryption key, including the release key that is used to
            sign the APK for publishing.
            You can define the keys for the debug and release targets in src/debug/ and src/release/.
        -->
        <meta-data
            android:name="com.google.android.geo.API_KEY"
            android:value="@string/google_maps_key" />
        <activity
            android:name=".MapsActivity"
            android:label="@string/title_activity_maps">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

مورد نظر یک فایل به نام `google_maps_api.xml` ایجاد کرده است. این فایل حاوی دستور العمل هایی در خصوص نحوه ایجاد `Google Maps API key` می باشد. پس از دریافت `API key` اپلیکیشن، این کلید را در فایل `google_maps_api.xml` وارد نمایید.

## تنظیم فایل `layout`

در این مثال از کلاس **MapFragment** استفاده می کنیم. محتوای فایل layout خود را به صورت زیر ویرایش نمایید.

```
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:map="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/map"
    android:name="com.google.android.gms.maps.SupportMapFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.vogella.android.maps.MapsActivity"
/>>
```

## ویرایش

بیاده سازی کلاس **activity** خود را به صورت زیر ویرایش نمایید.

```
package com.vogella.android.maps;
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.MapFragment;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;
public class MainActivity extends Activity {
    static final LatLng HAMBURG = new LatLng(53.558, 9.927);
    static final LatLng KIEL = new LatLng(53.551, 9.993);
    private GoogleMap map;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        map = ((MapFragment) getFragmentManager().findFragmentById(R.id.map))
            .getMap();
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.activity_main, menu);
        return true;
    }
}
```

**توجه:** برای اجرای این مثال لازم است قابلیت پشتیبانی از multidex را فعال نمایید.

```
android {
```

```
compileSdkVersion 25
buildToolsVersion "24.0.0"
defaultConfig {
    applicationId "com.vogella.android.maps"
    minSdkVersion 23
    targetSdkVersion 25
    versionCode 1
    versionName "1.0"
    multiDexEnabled true
    testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
}
buildTypes {
    release {
        minifyEnabled false
        proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
    }
}
```

## اجرا و تست اپلیکیشن

اپلیکیشن را اجرا کرده و تست نمایید. با توجه به منطق برنامه باید بتوانید بر روی Map حرکت کرده و قسمت هایی از آن بزرگ نمایی/کوچک نمایی کنید.



865

آدرس آموزشگاه : تهران - خیابان شریعتی - بالاتر از خیابان ملک - جنب بانک صادرات - پلاک 651 طبقه دوم - واحد 7

88146323 - 88446780 - 88146330



866

آدرس آموزشگاه : تهران - خیابان شریعتی - بالاتر از خیابان ملک - جنب بانک صادرات - پلاک 651 طبقه دوم - واحد 7

88146323 - 88446780 - 88146330

## بخش اول :

# توابع کتابخانه ای اندروید مربوط به امکان سخت افزاری

## دوربین / Camera API

این مبحث به شرح نحوه‌ی استفاده از امکان سخت افزاری Camera از طریق آبجکت Intent و توابع کتابخانه‌ی اندروید API می‌پردازد. پروژه‌ی این آموزش در ویرایش 4.2 محیط کاری Eclipse نوشته شده و مبتنی بر ورژن 1.6 جاوا و نسخه‌ی 4.2 سیستم عامل اندروید می‌باشد.

### امکان سخت افزاری Camera

امروزه اغلب دستگاه‌های اندرویدی حداقل یک دوربین را دارند و برخی نیز یک دوربین در جلو و یک دوربین در عقب دستگاه دارند.

جهت استفاده از امکان سخت افزاری camera دستگاه اندرویدی، کافی است از اپلیکیشن آماده‌ی camera استفاده نمایید. بدین معنی که توسعه دهنده به واسطه‌ی یک آبجکت intent اپلیکیشن آماده‌ی Camera را اجرا کرده و سپس با خواندن داده‌های بازگشته اپلیکیشن به نتیجه دست پیدا می‌کند.

روش دیگری که می‌توان مورد استفاده قرار داد، این است که camera یا امکان سخت افزاری دوربین را با فراخوانی توابع کتابخانه‌ی Camera API مستقیماً در بطن پروژه جاسازی نمایید.

مبحث زیر مثال‌های کاربردی درباره‌ی پیاده سازی هر دو روش ارائه می‌دهد.

### آموزش: اجرای اپلیکیشن Camera و گرفتن یک عکس با فراخوانی آبجکت Intent

یک پروژه و activity جدید به ترتیب به نام‌های de.vogella.android.imagepick و ImagePickActivity ایجاد نمایید.

یک المان ImageView که مقدار ID آن بر روی result تنظیم شده در لایه XML اپلیکیشن (فایل layout)، ایجاد نمایید. سپس یک المان button با ویژگی onClick (property) که به متدهای اشاره دارد، در این فایل اضافه نمایید.

حال کلاس ImagePickActivity را به صورت زیر ویرایش نمایید.

```
package de.vogella.android.imagepick;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import android.app.Activity;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.Bundle;
import android.view.View;
import android.widget.ImageView;
public class ImagePickActivity extends Activity {
    private static final int REQUEST_CODE = 1;
    private Bitmap bitmap;
    private ImageView imageView;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        imageView = (ImageView) findViewById(R.id.result);
    }
    public void onClick(View View) {
        Intent intent = new Intent();
        intent.setType("image/*");
        intent.setAction(Intent.ACTION_GET_CONTENT);
        intent.addCategory(Intent.CATEGORY_OPENABLE);
        startActivityForResult(intent, REQUEST_CODE);
    }
    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        InputStream stream = null;
        if (requestCode == REQUEST_CODE && resultCode == Activity.RESULT_OK)
            try {
                // recycle unused bitmaps
                if (bitmap != null) {
                    bitmap.recycle();
                }
                stream = getContentResolver().openInputStream(data.getData());
                bitmap = BitmapFactory.decodeStream(stream);
                imageView.setImageBitmap(bitmap);
            } catch (FileNotFoundException e) {
                e.printStackTrace();
            }
    }
}
```

```

        } finally ch (IOException e) {
            e{
                if (stream != null)
                    try {
                        stream.close();
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
            }
        }
    }
}

```

## آموزش: پیاده سازی توابع کتابخانه ای Camera API

در تمرین جاری، یک اپلیکیشن طراحی می کنید که به کاربر امکان می دهد با استفاده از دوربین جلوی دستگاه عکس تهیه کرده و بعد آن را بر روی حافظه ای خارجی (SD) ذخیره نماید.

**توجه:** در صورت استفاده از نرم افزار شبیه ساز محیط اندروید (Android emulator)، لازم است امکان سخت افزاری camera را تنظیم کرده و یک حافظه ای خارجی (SD card) به هنگام ایجاد دستگاه مجازی اندروید تعریف کرده باشد.

یک پروژه ای اندروید و activity جدید به ترتیب به نام های de.vogella.camera.api و de.vogella.camera.api MakePhotoActivity ایجاد نمایید.

لازم است مجوز android.permission.CAMERA را برای دسترسی به امکان سخت افزاری دوربین و android.permission.WRITE\_EXTERNAL\_STORAGE برای ذخیره ای اطلاعات بر روی حافظه ای خارجی به فایل AndroidManifest.xml اضافه نمایید.

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="de.vogella.cameara.api"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk android:minSdkVersion="15" />
    <uses-permission android:name="android.permission.CAMERA"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:name="de.vogella.camera.api.MakePhotoActivity"
            android:label="@string/app_name" >

```

```

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

```

فایل main.xml مقیم در پوشه `res/layout` را به صورت زیر ویرایش نمایید.

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
    <Button
        android:id="@+id/captureFront"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:onClick="onClick"
        android:text="Make Photo" />
</RelativeLayout>

```

یک کلاس به نام `PhotoHandler` را به صورت زیر اعلان و پیاده سازی کنید که وظیفه آن جایگذاری فایل تصویری تهیه شده در حافظه خارجی (SD card) می باشد.

```

package de.vogella.camera.api;
import java.io.File;
import java.io.FileOutputStream;
import java.text.SimpleDateFormat;
import java.util.Date;
import android.content.Context;
import android.hardware.Camera;
import android.hardware.Camera.PictureCallback;
import android.os.Environment;
import android.util.Log;
import android.widget.Toast;
public class PhotoHandler implements PictureCallback {
    private final Context context;
    public PhotoHandler(Context context) {
        this.context = context;
    }
    @Override
    public void onPictureTaken(byte[] data, Camera camera) {
        File pictureFileDir = getDir();
        if (!pictureFileDir.exists() && !pictureFileDir.mkdirs()) {
            Log.d(MakePhotoActivity.DEBUG_TAG, "Can't create directory to save image.");
            Toast.makeText(context, "Can't create directory to save image.",
                    Toast.LENGTH_LONG).show();
        }
    }
}

```

```

        return;
    }
    SimpleDateFormat dateFormat = new SimpleDateFormat("yyyyMMddhhmmss");
    String date = dateFormat.format(new Date());
    String photoFile = "Picture_" + date + ".jpg";
    String filename = pictureFileDir.getPath() + File.separator + photoFile;
    File pictureFile = new File(filename);
    try {
        FileOutputStream fos = new FileOutputStream(pictureFile);
        fos.write(data);
        fos.close();
        Toast.makeText(context, "New Image saved:" + photoFile,
                Toast.LENGTH_LONG).show();
    } catch (Exception error) {
        Log.d(MakePhotoActivity.DEBUG_TAG, "File" + filename + "not saved:"
                + error.getMessage());
        Toast.makeText(context, "Image could not be saved.",
                Toast.LENGTH_LONG).show();
    }
}
private File getDir() {
    File sdDir = Environment
            .getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES);
    return new File(sdDir, "CameraAPIDemo");
}
}

```

محتوای کلاس MakePhotoActivity را به صورت زیر ویرایش نمایید. این کلاس فرض را بر این می گذارد که شما یک المان button متناظر در لایه ای xml، فایل layout خود تعریف کرده اید که ویژگی onClick آن به متده متند onClick() در کلاس activity اشاره می کند.

```

package de.vogella.camera.api;
import android.app.Activity;
import android.content.pm.PackageManager;
import android.hardware.Camera;
import android.hardware.Camera.CameraInfo;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Toast;
import de.vogella.cameara.api.R;
public class MakePhotoActivity extends Activity {
    private final static String DEBUG_TAG = "MakePhotoActivity";
    private Camera camera;
    private int camerald = 0;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        // do we have a camera?

```

```

if (!getPackageManager()
    .hasSystemFeature(PackageManager.FEATURE_CAMERA)) {
    Toast.makeText(this, "No camera on this device", Toast.LENGTH_LONG)
        .show();
} else {
    camerald = findFrontFacingCamera();
    if (camerald < 0) {
        Toast.makeText(this, "No front facing camera found.",
            Toast.LENGTH_LONG).show();
    } else {
        camera = Camera.open(camerald);
    }
}
public void onClick(View view) {
    camera.startPreview();
    camera.takePicture(null, null,
        new PhotoHandler(getApplicationContext()));
}
private int findFrontFacingCamera() {
    int camerald = -1;
    // Search for the front facing camera
    int numberofCameras = Camera.getNumberOfCameras();
    for (int i = 0; i < numberofCameras; i++) {
        CameralInfo info = new CameralInfo();
        Camera.getCameralInfo(i, info);
        if (info.facing == CameralInfo.CAMERA_FACING_FRONT) {
            Log.d(DEBUG_TAG, "Camera found");
            camerald = i;
            break;
        }
    }
    return camerald;
}
@Override
protected void onPause() {
    if (camera != null) {
        camera.release();
        camera = null;
    }
    super.onPause();
}
}

```



بخش دوم :

## آموزش استفاده از Android Sensor Manager

آموزش حاضر به شرح نحوه استفاده از کلاس SensorManager می‌پردازد. پروژه‌ی این مبحث در ویرایش 3.6 محیط کاری Eclipse نوشته شده و بر پایه‌ی ورژن 1.6 زبان جاوا و نسخه‌ی 2.3.3 اندروید (Gingerbread) می‌باشد.

پیاده سازی کلاس SensorManager و دسترسی به حسگرهای دستگاه اندروید

برای دسترسی به حسگرهای دستگاه اندروید می بایست نمونه ای از کلاس SensorManager را پیاده سازی نمایید. از جمله‌ی این حسگرهای می توان به شتاب سنج یا accelerometer اشاره کرد. متاسفانه، امکان تست شتاب سنج و شبیه سازی آن در Android emulator وجود ندارد.

برای دسترسی به کلاس SensorManager کافی است متدهای getSystemService(SENSOR\_SERVICE) را فراخوانی نمایید. کلاس Sensor ثوابت متعددی برای دسترسی به حسگرهای مختلف فراهم می آورد (constants).

- Sensor.TYPE\_GYROSCOPE
- Sensor.TYPE\_MAGNETIC\_FIELD
- Sensor.TYPE\_ORIENTATION
- Sensor.TYPE\_ACCELEROMETER

می توانید با فراخوانی متدهای sensorManager.getDefaultSensor() به حسگر مربوطه دسترسی داشته باشید. این متدهای نوع حسگر (sensor type) و میزان زمان غیرفعال بودن تا فراخوانی حسگر را که به عنوان ثابت در سطح کلاس SensorManager تعریف شده، به عنوان پارامتر می پذیرد.

## گوش فرادهنده به تغییرات حسگر (Sensor listener)

پس از دسترسی به حسگر مربوطه، لازم است یک SensorEventListener برای آن تعریف نمایید. این listener به تغییرات مرتبط با حسگر (sensor) گوش می دهد و به محض رخداد تغییر در اطلاعات حسگر مورد نظر، از آن با خبر می شود.

به منظور جلوگیری از انلاف باتری، می توانید listener خود را در متدهای onResume() اعلان و ثبت نموده و داخل متدهای onPause() از حالت مزبور خارج نمایید (deregister کنید).

## آموزش استفاده ای کاربردی از حسگر Accelerometer

در این بخش یک اپلیکیشن خواهیم نوشت که در صورت تکان خوردن دستگاه، به صورت خودکار رنگ پس زمینه را تغییر می دهد. برای این منظور ابتدا یک پروژه ای اندرویدی و activity به نام های SensorTestActivity و de.vogella.android.sensor ایجاد نمایید.

محتوای فایل layout خود را به صورت زیر ویرایش نمایید.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="Shake to get a toast and to switch color" />
</LinearLayout>
```

حال کد کلاس activity خود را به صورت زیر ویرایش نمایید.

```
package de.vogella.android.sensor;
import android.app.Activity;
import android.graphics.Color;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.widget.Toast;
public class SensorTestActivity extends Activity implements SensorEventListener {
    private SensorManager sensorManager;
    private boolean color = false;
    private View view;
    private long lastUpdate;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
            WindowManager.LayoutParams.FLAG_FULLSCREEN);
        super.onCreate(savedInstanceState);
```

```

        setContentView(R.layout.main);
        view = findViewById(R.id.textView);
        view.setBackgroundColor(Color.GREEN);
        sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
        lastUpdate = System.currentTimeMillis();
    }

    @Override
    public void onSensorChanged(SensorEvent event) {
        if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
            getAccelerometer(event);
        }
    }

    private void getAccelerometer(SensorEvent event) {
        float[] values = event.values;
        // Movement
        float x = values[0];
        float y = values[1];
        float z = values[2];
        float accelerationSquareRoot = (x * x + y * y + z * z)
            / (SensorManager.GRAVITY_EARTH * SensorManager.GRAVITY_EARTH);
        long actualTime = event.timestamp;
        if (accelerationSquareRoot >= 2) //
        {
            if (actualTime - lastUpdate < 200) {
                return;
            }
            lastUpdate = actualTime;
            Toast.makeText(this, "Device was shuffled", Toast.LENGTH_SHORT)
                .show();
            if (color) {
                view.setBackgroundColor(Color.GREEN);
            } else {
                view.setBackgroundColor(Color.RED);
            }
            color = !color;
        }
    }

    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {
    }

    @Override
    protected void onResume() {
        super.onResume();
        // register this class as a listener for the orientation and
        // accelerometer sensors
        sensorManager.registerListener(this,
            sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER),
            SensorManager.SENSOR_DELAY_NORMAL);
    }

    @Override
    protected void onPause() {
        // unregister listener
    }
}

```

```

        super.onPause();
        sensorManager.unregisterListener(this);
    }
}

```

## آموزش کاربردی: ساخت یک قطب نما

یک پروژه و activity جدید اندرویدی به ترتیب به نام های de.vogella.android.sensor.compass و MainActivity ایجاد نمایید.

اکنون یک کلاس View با پیاده سازی اختصاصی به صورت زیر ایجاد کنید.

```

package de.vogella.android.sensor.compass;
import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.view.View;
public class MyCompassView extends View {
    private Paint paint;
    private float position = 0;
    public MyCompassView(Context context) {
        super(context);
        init();
    }
    private void init() {
        paint = new Paint();
        paint.setAntiAlias(true);
        paint.setStrokeWidth(2);
        paint.setTextSize(25);
        paint.setStyle(Paint.Style.STROKE);
        paint.setColor(Color.WHITE);
    }
    @Override
    protected void onDraw(Canvas canvas) {
        int xPoint = getMeasuredWidth() / 2;
        int yPoint = getMeasuredHeight() / 2;
        float radius = (float) (Math.max(xPoint, yPoint) * 0.6);
        canvas.drawCircle(xPoint, yPoint, radius, paint);
        canvas.drawRect(0, 0, getMeasuredWidth(), getMeasuredHeight(), paint);
        // 3.143 is a good approximation for the circle
        canvas.drawLine(
            xPoint,
            yPoint,
            (float) (xPoint + radius
                * Math.sin((double) (-position) / 180 * 3.143)),
            (float) (yPoint - radius
                * Math.cos((double) (-position) / 180 * 3.143)), paint);
    }
}

```

```

        canvas.drawText(String.valueOf(position), xPoint, yPoint, paint);
    }
    public void updateData(float position) {
        this.position = position;
        invalidate();
    }
}

```

بدنه ی کلاس activity پروژه خود را به صورت زیر ویرایش کنید.

```

package de.vogella.android.sensor.compass;
import android.app.Activity;
import android.content.Context;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.util.Log;
import android.widget.Toast;
public class MainActivity extends Activity {
    private static SensorManager sensorService;
    private MyCompassView compassView;
    private Sensor sensor;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        compassView = new MyCompassView(this);
        setContentView(compassView);
        sensorService = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
        sensor = sensorService.getDefaultSensor(Sensor.TYPE_ORIENTATION);
        if (sensor != null) {
            sensorService.registerListener(mySensorEventListener, sensor,
                SensorManager.SENSOR_DELAY_NORMAL);
            Log.i("Compass MainActivity", "Registered for ORIENTATION Sensor");
        } else {
            Log.e("Compass MainActivity", "Registered for ORIENTATION Sensor");
            Toast.makeText(this, "ORIENTATION Sensor not found",
                Toast.LENGTH_LONG).show();
        }
        finish();
    }
    private SensorEventListener mySensorEventListener = new SensorEventListener() {
        @Override
        public void onAccuracyChanged(Sensor sensor, int accuracy) {
        }
        @Override
        public void onSensorChanged(SensorEvent event) {
            // angle between the magnetic north direction
            // 0=North, 90=East, 180=South, 270=West
            float azimuth = event.values[0];

```

```
        compassView.updateData(azimuth);
    }
};

@Override
protected void onDestroy() {
    super.onDestroy();
    if (sensor != null) {
        sensorService.unregisterListener(mySensorEventListener);
    }
}
}
```



## بخش سوم :

# آموزش استفاده ای کاربردی از قابلیت single touch و multi touch در اندروید / آموزش پیاده سازی و استفاده از توابع کتابخانه ای touch API در اپلیکیشن های اندرویدی

آموزش حاضر به شرح پیاده سازی و استفاده از توابع کتابخانه ای touch API و قابلیت لمس نمایشگر در اپلیکیشن های اندرویدی می پردازد.

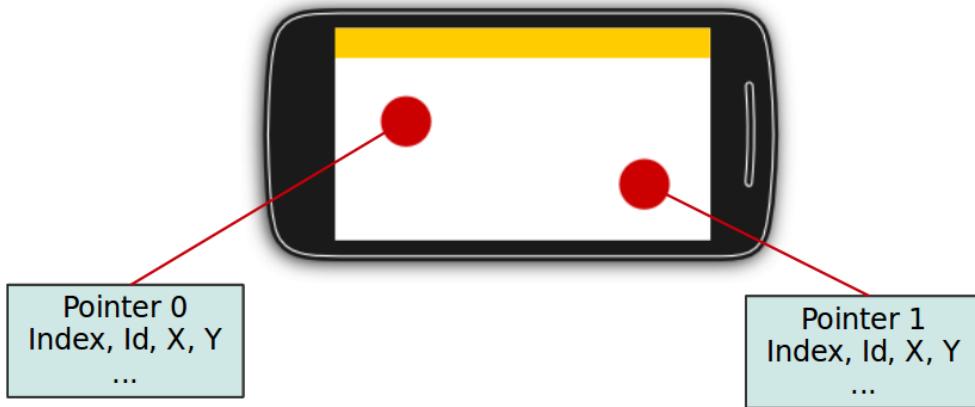
## رخدادهای مربوط به لمس نمایشگر (Android touch)

واکنش نشان دادن به event های مربوط به touch

کلاس استاندارد و پرکاربرد View کتابخانه های اندروید به راحتی قادر است از رخدادهای touch پشتیبانی کرده و به آن ها پاسخ دهد.

می توانید به رخدادهایی که توسط لمس نمایشگر فعال می شوند، در view و activity های view pointer که پروژه ای خود پاسخ دهید. سیستم اندروید این قابلیت را دارد تا همزمان از چندین نشانگر انگشت بر روی نمایشگر و تعامل کاربر با چند نقطه از صفحه نمایش دستگاه می باشد، پشتیبانی کند.

# Pointers



کلاس پایه ای (پدر) که برای پشتیبانی از قابلیت لمس نمایشگر، توسعه می باشد از آن ارث بری کند، کلاس MotionEvent می باشد. این کلاس به واسطه ای متده onTouchEvent() به عنوان پارامتر به آبجکت های Views فرستاده می شود.

به منظور فراهم آوردن قابلیت پاسخ دهی و واکنش نشان دادن به event های مربوط به touch (لمس نمایشگر)، کافی است متده onTouchEvent() را بازنویسی (override) نمایید.

کلاس MotionEvent تمامی اطلاعات مربوط به قابلیت touch نظیر تعداد pointer ها، مختصات Y/X، میزان اندازه و pressure هر pointer را در بردارد.

چنانچه event فعال شده توسط لمس نمایشگر را view مدیریت کند، آنگاه متده name برده مقدار بولی true را در خروجی برمی گرداند. لازم به ذکر است که سیستم اندروید تلاش می کند تودرتوترین (deepest) کلاس view ای که event مربوط به touch را مدیریت کرده و مقدار view را برگرداند، پیدا کند. در صورتی که view خود داخل view دیگر (parent view) یا میزبان (mzban) جای گرفته باشد، آنگاه view میزبان می تواند با بازگردانی true ازتابع onInterceptTouchEvent() ارسال رخداد به view زیرمجموعه را بگیرد. در نتیجه ای

عملیات، رخداد view ای که قبلاً قرار بود رخداد اصلی را دریافت کند، ارسال شده و view میزبان خود مدیریت event را بر عهده می‌گیرد.

برای واکنش نشان دادن به رخدادهای لمس نمایشگر در سطح یک activity، کافی است View های مربوطه اعلان نمایید.

### وضعیت لمس یک نقطه از نمایشگر (single touch)

چنانچه تنها یک انگشت در آن واحد با سطح نمایشگر برخورد داشته و سیستم اندروید تنها یک ورودی دریافت کرده باشد، در آن صورت توسعه دهنده می‌تواند با فراخوانی متد `getX()` و `getY()`، مقادیر و اطلاعات مربوط به موقعیت فعلی (مختصات) اولین انگشتی که با صفحه تعامل داشته را بازیابی کند.

با فراخوانی متد `getAction()` توسعه دهنده action و فعلی که اجرا شده را در خروجی دریافت می‌کند.

کلاس MotionEvent از مجموعه کتابخانه های اندروید تعدادی ثابت (constant) در اختیار برنامه نویس قرار می‌دهد که از طریق آن ها قادر خواهد بود اطلاعات مربوط به action رخداد را مشخص نماید.

Event	شرح
<code>MotionEvent.ACTION_DOWN</code>	<b>New touch started</b> (انگشت با سطح نمایشگر برخورد کرده و در نتیجه touch یا لمس دیگری آغاز شده است)
<code>MotionEvent.ACTION_MOVE</code>	<b>Finger is moving</b> (انگشت کاربر در سطح نمایشگر در حال حرکت است)
<code>MotionEvent.ACTION_UP</code>	<b>Finger went up</b> (انگشت کاربر دیگر با نمایشگر تعامل ندارد)

Event	شرح
MotionEvent.ACTION_CANCEL	رخداد جاری لغو شده (به view زیرمجموعه یا فرزند ارسال نمی شود) و دیگری view مدیریت event را بر عهده می گیرد.
MotionEvent.ACTION_POINTER_DOWN	Pointer down (multi-touch) انگشت کاربر با سطح نمایشگر برخورد کرده و بر روی آن قرار دارد (وضعیت پشتیبانی از لمس همزمان چند نقطه از نمایشگر).
MotionEvent.ACTION_POINTER_UP	Pointer up (multi-touch) انگشت کاربر از روی سطح نمایشگر برداشته شده است (وضعیت پشتیبانی از لمس همزمان چند نقطه از نمایشگر).

### (Multi touch)

قابلیت پیاده سازی و پشتیبانی از وضعیت لمس چندین نقطه از نمایشگر به طور همزمان در اپلیکیشن های اندرویدی از ویرایش 2.0 کتابخانه های اندروید اضافه شده و ورژن 2.2 به مراتب ارتقا یافت.

MotionEvent.ACTION\_POINTER\_UP و MotionEvent.ACTION\_POINTER\_DOWN زمانی ارسال می شوند که انگشت دوم با سطح نمایشگر تعامل برقرار کند.

دو رخداد انگشت اول با سطح نمایشگر مورد استفاده قرار می گیرند.

متدهای getPointerCount() که در سطح کلاس MotionEvent موجود است، به شما این اجازه را می دهد تا تعداد pointer ها (انگشت یا شی دیگری که با نمایشگر ارتباط داشته) را بازیابی نمایید. تمامی رخدادهای مربوطه، به همراه موقعیت و مختصات pointer ها همگی در نمونه ای ایجاد شده از MotionEvent که به عنوان پارامتر متدهای onTouch() دریافت می کنید، موجود می باشد.

جهت رصد (track) و ردیابی اطلاعات مربوط به تمامی pointer ها، لازم است دو متند MotionEvent.getActionMasked() و MotionEvent.getActionIndex() را فراخوانی کنید و از طریق آن ها اندیس pointer و رخداد مربوط به touch که از برخورد این pointer فعال شده، شناسایی نمایید.

اندیس pointer ممکن است در طول زمان تغییر کند. به طور مثال می‌توان به زمانی اشاره کرد که یک انگشت از سطح نمایشگر برداشته می‌شود. نسخه‌ی پایدار از pointer id همان MotionEvent.getPointerId(pointerIndex) است که با متند getPointerId(pointerIndex) قابل شناسایی و بازیابی می‌باشد.

کاربرد آن به صورت عملی در زیر نمایش داده شده است.

```
@Override
public boolean onTouchEvent(MotionEvent event) {
    // get pointer index from the event object
    int pointerIndex = event.getActionIndex();
    // get pointer ID
    int pointerId = event.getPointerId(pointerIndex);
    // get masked (not specific to a pointer) action
    int maskedAction = event.getActionMasked();
    switch (maskedAction) {
        case MotionEvent.ACTION_DOWN:
        case MotionEvent.ACTION_POINTER_DOWN:
            // TODO use data
            break;
    }
    case MotionEvent.ACTION_MOVE: // a pointer was moved
        // TODO use data
        break;
    }
    case MotionEvent.ACTION_UP:
    case MotionEvent.ACTION_POINTER_UP:
    case MotionEvent.ACTION_CANCEL:
        // TODO use data
        break;
    }
    }
    invalidate();
    return true;
}
```

**نکته:** امکان تست قابلیت Multitouch در شبیه ساز وجود ندارد. برای آزمودن قابلیت لمس چندین نقطه از نمایشگر به طور همزمان، نیاز به یک دستگاه واقعی اندروید دارید (دستگاهی که ورودی را دریافت می کند باید یک دستگاه حقیقی اندروید باشد).

## کلاس GestureDetectors

چارچوب نرم افزاری اندروید کلاسی به نام GestureDetector را ارائه می دهد که از اطلاعات و اعضای آبجکت (نمونه ی ساخته شده از کلاس) MotionEvents استفاده می نماید. این کلاس قادر است انواع حرکت ها (gesture) و رخدادها (event ها) را با استفاده از نمونه ی کلاس MotionEvents که به عنوان پارامتر ارائه می شود، شناسایی کند. تابع بازفرارخوان GestureDetector.OnGestureListener (رخدادی که بر اثر حرکت فعال می شود) مورد نظر مطلع می سازد. motion event به طور مثال کلاس ScaleGestureDetector این امکان را به شما می دهد تا حرکت از پیش تعیین شده ی کوچک و بزرگ کردن یک آبجکت با دو انگشت را تعریف نمایید.

## تمرین: پیاده سازی view اختصاصی و مدیریت رخدادهای مربوط به لمس یک نقطه از نمایشگر (single touch event)

### ترسیم با استفاده از قابلیت لمس نمایشگر

تمرین حاضر نحوه ی مدیریت رخدادهای مربوط به لمس یک نقطه از نمایشگر را داخل یک view اختصاصی نمایش می دهد.

ابتدا یک پروژه و activity جدید به ترتیب به نام های com.vogella.android.touch.single و com.vogella.android.activity ایجاد نمایید. این activity از کلاس DialogFragment برای تنظیم رنگ قابل استفاده برای ترسیم در سطح نمایشگر بهره می گیرد.

حال یک فایل layout به نام fragment\_colorpicker.xml با محتوای زیر ایجاد نمایید.

```
<?xml version="1.0" encoding="utf-8"?>
```

```

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">
    <View
        android:layout_width="match_parent"
        android:layout_height="40dp"
        android:layout_margin="20dp"
        android:background="#ff0000"
        android:id="@+id/preview"
        />
    <SeekBar
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/red"
        android:progress="255"
        android:max="255"/>
    <SeekBar
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/green"
        android:max="255"/>
    <SeekBar
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/blue"
        android:max="255"/>
</LinearLayout>

```

کلاسی به نام TouchEventView تعریف کرده که یک آبجکت View با قابلیت پشتیبانی از لمس یک نقطه از نمایشگر در آن واحد را پیاده سازی می کند.

```

package com.vogella.android.touch.single;
import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Path;
import android.util.AttributeSet;
import android.util.Log;
import android.view.GestureDetector;
import android.view.MotionEvent;
import android.view.VelocityTracker;
import android.view.View;
public class TouchEventView extends View {
    private Paint paint = new Paint();
    private Path path = new Path();
    Context context;
    GestureDetector gestureDetector;
    public TouchEventView(Context context, AttributeSet attrs) {
        super(context, attrs);
        gestureDetector = new GestureDetector(context, new GestureListener());
    }
}

```

```

this.context = context;
paint.setAntiAlias(true);
paint.setStrokeWidth(6f);
paint.setColor(Color.BLACK);
paint.setStyle(Paint.Style.STROKE);
paint.setStrokeJoin(Paint.Join.ROUND);
}
public void setColor(int r, int g, int b) {
    int rgb = Color.rgb(r, g, b);
    paint.setColor(rgb);
}
private class GestureListener extends GestureDetector.SimpleOnGestureListener {
    // event when double tap occurs
    @Override
    public boolean onDoubleTap(MotionEvent e) {
        float x = e.getX();
        float y = e.getY();
        // clean drawing area on double tap
        path.reset();
        Log.d("Double Tap", "Tapped at: (" + x + "," + y + ")");
        return true;
    }
    @Override
    protected void onDraw(Canvas canvas) {
        canvas.drawPath(path, paint);
    }
    @Override
    public boolean onTouchEvent(MotionEvent event) {
        float eventX = event.getX();
        float eventY = event.getY();
        switch (event.getAction()) {
            case MotionEvent.ACTION_DOWN:
                path.moveTo(eventX, eventY);
                return true;
            case MotionEvent.ACTION_MOVE:
                path.lineTo(eventX, eventY);
                break;
            case MotionEvent.ACTION_UP:
                break;
            default:
                return false;
        }
        // for demonstration purposes
        gestureDetector.onTouchEvent(event);
        // Schedules a repaint.
        invalidate();
        return true;
    }
}

```

محتوای فایل activity\_main.xml را به صورت زیر اصلاح نمایید.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
<com.vogella.android.touch.single.SingleTouchEventView
    android:layout_width="match_parent"
    android:background="@drawable/dwarf"
    android:layout_height="match_parent"
    android:id="@+id/drawingview">
</com.vogella.android.touch.single.SingleTouchEventView>
</LinearLayout>
```

حال view مورد نظر را به کلاس activity خود اضافه نمایید.

```
package com.vogella.android.touch.single;
import android.app.Activity;
import android.os.Bundle;
public class SingleTouchActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(new SingleTouchEventView(this, null));
    }
}
```

اکنون زمانی که اپلیکیشن را اجرا می کنید، با استیتی با توجه به کدی که نوشته اید قادر باشید با انگشت خود بر روی نمایشگر دستگاه اشکال مختلفی ترسیم کنید (در محیط شبیه ساز با استفاده از موس).

کد اپلیکیشن خود را طوری ویرایش کنید که از تعریف و محتوای یک فایل layout با فرمات XML برای تنظیم ظاهر خود استفاده کند.

**نکته:** به منظور اعلان view دلخواه خود در لایه‌ی XML اپلیکیشن، کد موجود در فایل layout لازم است اسم و آدرس دقیق کلاس از جمله اطلاعات پکیج آن کلاس را ذکر نمایید.

## افزودن قابلیت تنظیم پهنه‌ی خط ترسیم (line width)

یک منو به اپلیکیشن خود اضافه کنید (المان مربوطه را در لایه‌ی XML، داخل فایل layout به صورت زیر اعلان نمایید).

```
<?xml version="1.0" encoding="utf-8"?>
```

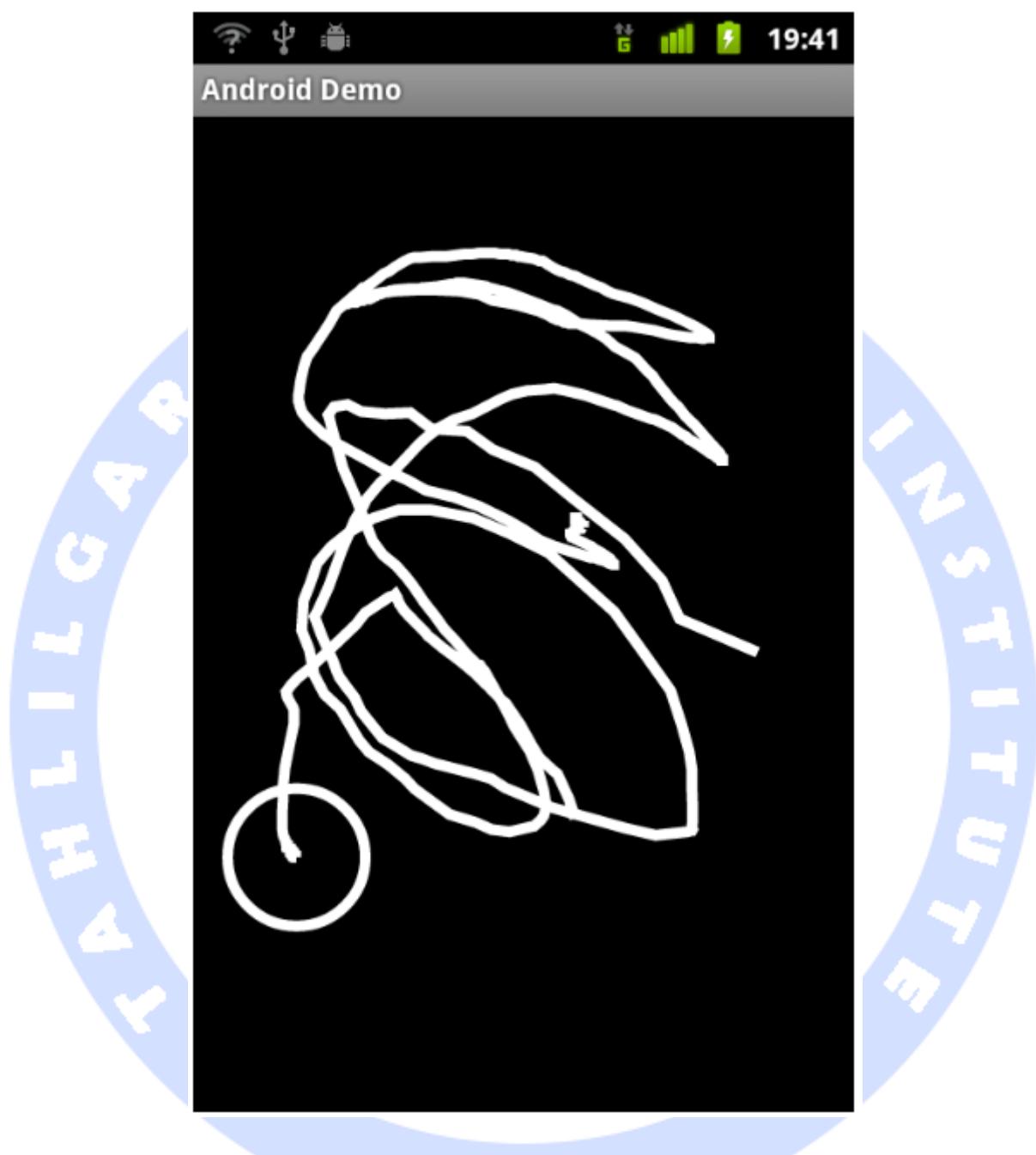
```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:title="Pick pen" android:id="@+id/action_pickpen"/>
</menu>
```

## رصد و ردیابی مختصات جاری انگشت کاربر بر روی نمایشگر

حال کدی به اپلیکیشن خود اضافه نمایید که موقعیت فعلی انگشت کاربر بر روی نمایشگر را با استفاده از یک دایره نشانه گذاری و دنبال کند. به منظور ترسیم دایره، می‌توانید متد Path.addCircle(x, y, 50, Path.Direction.CW) را بر روی نمونه‌ی ساخته شده از کلاس canvas فراخوانی کنید یا المان canvas را مستقیماً بکار ببرید.

اطمینان حاصل کنید که تنها موقعیت جاری انگشت کاربر بر روی نمایشگر با استفاده از شکل دایره نشانه گذاری و هایلایت می‌شود. شکل مجبور بایستی به مجرد برخورد انگشت کاربر با سطح نمایشگر دستگاه پدیدار شده و زمانی که تماس انگشت با صفحه قطع شد، محو گردد. خروجی می‌بایست شما به زیر باشد.





## تمرین: پیاده سازی قابلیت پشتیبانی از لمس چندین نقطه از نمایشگر (Multitouch)

در تمرین جاری، توسعه دهنده می باشد یک view اعلان کند که از قابلیت multitouch پشتیبانی کرده و از این طریق امکان ردیابی و پاسخ دهی به چندین انگشت بر روی نمایشگر به

890

آدرس آموزشگاه : تهران - خیابان شریعتی - بالاتر از خیابان ملک - جنب بانک صادرات - پلاک 651 طبقه دوم - واحد 7

88146323 - 88446780 - 88146330

طور همزمان را فراهم آورد. لازم به ذکر است که تمرین پیاده سازی قابلیت لمس چند نقطه از نمایشگر در آن واحد در بستر شبیه ساز ممکن نبوده و توسعه دهنده می بایست برای این منظور حتما از دستگاه واقعی اندروید استفاده کند.

همان طور که قبل گفته شد، نرم افزار emulator اندروید تنها اجزه ی شبیه سازی را (با اشاره گر موس) برای برنامه نویس فراهم می کند.

یک پروژه و activity جدید به ترتیب به نام های com.vogella.android.multitouch و com.vogella.android.multitouch ایجاد نمایید.

ابتدا کلاسی به نام MultitouchView را به صورت زیر پیاده سازی کنید.

```
package com.vogella.android.multitouch;
import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.PointF;
import android.util.AttributeSet;
import android.util.SparseArray;
import android.view.MotionEvent;
import android.view.View;
public class MultitouchView extends View {
    private static final int SIZE = 60;
    private SparseArray<PointF> mActivePointers;
    private Paint mPaint;
    private int[] colors = { Color.BLUE, Color.GREEN, Color.MAGENTA,
        Color.BLACK, Color.CYAN, Color.GRAY, Color.RED, Color.DKGRAY,
        Color.LTGRAY, Color.YELLOW };
    private Paint textPaint;
    public MultitouchView(Context context, AttributeSet attrs) {
        super(context, attrs);
        initView();
    }
    private void initView() {
        mActivePointers = new SparseArray<PointF>();
        mPaint = new Paint(Paint.ANTI_ALIAS_FLAG);
        // set painter color to a color you like
        mPaint.setColor(Color.BLUE);
        mPaint.setStyle(Paint.Style.FILL_AND_STROKE);
        textPaint = new Paint(Paint.ANTI_ALIAS_FLAG);
        textPaint.setTextSize(20);
    }
    @Override
    public boolean onTouchEvent(MotionEvent event) {
```

```

// get pointer index from the event object
int pointerIndex = event.getActionIndex();
// get pointer ID
int pointerId = event.getPointerId(pointerIndex);
// get masked (not specific to a pointer) action
int maskedAction = event.getActionMasked();
switch (maskedAction) {
    case MotionEvent.ACTION_DOWN:
    case MotionEvent.ACTION_POINTER_DOWN:
        // We have a new pointer. Lets add it to the list of pointers
        PointF f = new PointF();
        f.x = event.getX(pointerIndex);
        f.y = event.getY(pointerIndex);
        mActivePointers.put(pointerId, f);
        break;
    }
    case MotionEvent.ACTION_MOVE: { // a pointer was moved
        for (int size = event.getPointerCount(), i = 0; i < size; i++) {
            PointF point = mActivePointers.get(event.getPointerId(i));
            if (point != null) {
                point.x = event.getX(i);
                point.y = event.getY(i);
            }
            break;
        }
    }
    case MotionEvent.ACTION_UP:
    case MotionEvent.ACTION_POINTER_UP:
    case MotionEvent.ACTION_CANCEL: {
        mActivePointers.remove(pointerId);
        break;
    }
}
invalidate();
return true;
}

@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    // draw all pointers
    for (int size = mActivePointers.size(), i = 0; i < size; i++) {
        PointF point = mActivePointers.valueAt(i);
        if (point != null)
            mPaint.setColor(colors[i % 9]);
        canvas.drawCircle(point.x, point.y, SIZE, mPaint);
    }
    canvas.drawText("Total pointers: " + mActivePointers.size(), 10, 40, textPaint);
}
}

```

زیر را به فایل layout متناظر activity خود اضافه نمایید.

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <com.vogella.android.multitouch.MultitouchView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
    />
</RelativeLayout>
```

کلاس activity ای که قبل ایجاد شده نیازی به تغییر ندارد.

```
package com.vogella.android.multitouch;
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.activity_main, menu);
        return true;
    }
}
```

اپلیکیشن در زمان اجرا به شما این امکان را می دهد تا با انگشت خود بر روی نمایشگر اشکال مختلف ترسیم نمایید. در هر دستگاهی، محدودیت از پیش تعیین شده ای برای حداکثر تعداد نقاط قابل پشتیبانی در آن واحد در نمایشگر وجود دارد. لازم است بررسی کنید دستگاه شما حداقل چه تعداد pointer را به طور همزمان پشتیبانی می کند. در پایان، اپلیکیشن بایستی ظاهری مشابه زیر داشته باشد.



یک پروژه و `de.vogella.android.touch.scaledetector` activity به ترتیب به نام های `ScaleDetectorTestActivity`

کلاس زیر را پیاده سازی نمایید.

```
package de.vogella.android.touch.scaledetector;
import android.content.Context;
import android.graphics.Canvas;
import android.graphics.drawable.Drawable;
import android.view.MotionEvent;
import android.view.ScaleGestureDetector;
import android.view.View;
public class ImageViewWithZoom extends View {
    private Drawable image;
    private float scaleFactor = 1.0f;
    private ScaleGestureDetector scaleGestureDetector;
    public ImageViewWithZoom(Context context) {
        super(context);
```

```

        image = context.getResources().getDrawable(R.drawable.icon);
        setFocusable(true);
        image.setBounds(0, 0, image.getIntrinsicWidth(),
                image.getIntrinsicHeight());
        scaleGestureDetector = new ScaleGestureDetector(context,
                new ScaleListener());
    }

    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        // Set the image boundaries
        canvas.save();
        canvas.scale(scaleFactor, scaleFactor);
        image.draw(canvas);
        canvas.restore();
    }

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        scaleGestureDetector.onTouchEvent(event);
        invalidate();
        return true;
    }

    private class ScaleListener extends
            ScaleGestureDetector.SimpleOnScaleGestureListener {
        @Override
        public boolean onScale(ScaleGestureDetector detector) {
            scaleFactor *= detector.getScaleFactor();
            // don't let the object get too small or too large.
            scaleFactor = Math.max(0.1f, Math.min(scaleFactor, 5.0f));
            invalidate();
            return true;
        }
    }
}

```

این View را به activity خود اضافه نمایید.

```

package de.vogella.android.touch.scaledetector;
import android.app.Activity;
import android.os.Bundle;
public class ScaleDetectorTestActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(new ImageViewWithZoom(this));
    }
}

```

زمانی که اپلیکیشن را راه اندازی می کنید، بایستی بتوانید با استفاده از قابلیت multi-touch تصویر زیر را کوچک و بزرگ نمایید (pinch zoom).



## بخش چهارم :

896

آدرس آموزشگاه : تهران - خیابان شریعتی - بالاتر از خیابان ملک - جنب بانک صادرات - پلاک 651 طبقه دوم - واحد 7

88146323 - 88446780 - 88146330

# آموزش کار با Gestures و مدیریت رخدادهای فعال شده از حرکات مختلف بر روی نمایشگر

مبخت حاضر نحوه ای استفاده از GestureOverlayView و Gestures در اندروید را به شما می آموزد. پروژه های این آموزش در نسخه ۳.۶ محیط کاری Eclipse و با استفاده از ویرایش ۱.۶ زبان جاوا نوشته شده و مبتنی بر ورژن ۲.۳ سیستم عامل اندروید (Gingerbread) می باشد.

## کار با GestureOverlayView ها و پیاده سازی کلاس Gesture در اندروید

سیستم اندروید، برای لمس و حرکت دادن انگشت بر روی سطح نمایشگر، رخدادهای معینی نظیر بزرگ نمایی، فشار دادن طولانی و پیمایش با نوار اسکرول در صفحه را ارائه می دهد. به این رخدادها که در بر اثر لمس نمایشگر اتفاق افتاده و فعال می شوند در اصطلاح gestures گفته می شود.

برای استفاده از این امکان اندروید، اپلیکیشن شما بایستی یک نمونه از کلاس GestureOverlayView را پیاده سازی کند. سپس می توانید در این view، سایر کامپوننت های رابط کاربری و view های خود را جایگذاری نمایید.

gesture ها را منابع باینری تشکیل داده و در اختیار توسعه دهنده قرار می دهند. این منابع را می توان با استفاده از Android SDK تهیه کرد.

جهت بارگذاری این منابع باینری و gesture ها در activity یا صفحه قابل مشاهده برای کاربر لازم است متده `GestureLib.fromRawResource()` را فراخوانی کنید. پس از آنکه gesture توسط سیستم شناسایی شد، متده `"onGesturePerformedListener"` فراخوانده می شود. اما قبل از آن کلاس activity ای که می خواهد gesture را بارگذاری کند باید توابع اینترفیس "OnGesturePerformedListener" را پیاده سازی کرده و با فراخوانی متده

"بر روی آبجکت ساخته شده از روی addOnGesturePerformedListener()" خود را معرفی نماید.

سیستم اندروید، آن دسته از gesture هایی را که شناخته با رنگ زرد و gesture هایی که برایش قابل شناسایی نیستند را با رنگ کم رنگ نمایش می دهد. توسعه دهنده، در صورت تمایل، می تواند این قابلیت را غیر فعال کند. برای نیل به این هدف تنها کافی است که یکی از دوتابع setUncertainGestureColor(Color.TRANSPARENT) یا setGestureColor(Color.TRANSPARENT) را بر روی آبجکت یا نمونه از GestureOverlayView فراخوانی نماید.

اگر از "GestureBuilder" برای ساخت gesture در شبیه ساز اندروید (Emulator) استفاده کنید، در آن صورت می توانید تعداد زیادی gesture با اسم یکسان ایجاد نمایید. از این طریق امکانی برای شما فراهم می شود تا gesture دلخواه و مد نظر خود را پیدا کرده و تعیین نمایید. اگر برای ویرایش 1.6 اندروید Android Emulator ایجاد کنید، این اپلیکیشن به صورت پیش فرض بر روی دستگاه شما نصب شده و آماده استفاده می باشد. لازم است یک دستگاه با قابلیت حافظه خارجی (sd card) ایجاد کنید چرا که در غیر این صورت امکان ذخیره ی gesture ها را نخواهید داشت. تمامی gesture ها در محیط شبیه ساز، داخل فایل gestures ذخیره می شوند.

شما می توانید تمامی gesture ها را به وسیله ی ابزار adb از شبیه ساز (emulator) به دستگاه محلی خود کپی نمایید. کافی است دستور زیر را در خط فرمان درج کنید:

```
./adb pull /sdcard/gestures ~/test
```

فایل gesture بایستی تحت آدرس "res/raw" در اپلیکیشن جایگذاری شود. پس از این کار می توانید از فایل مزبور به راحتی در آبجکت GestureOverlayView استفاده نمایید.

## مثال

یک پروژه و activity جدید به ترتیب به نام های "de.vogella.android.gestures" و "GestureTest" ایجاد نمایید. سپس تعدادی gesture ایجاد کرده و آن ها را طبق توضیح فوق، در اپلیکیشن مورد نظر جایگذاری نمایید.



متاسفانه GestureOverlayView از UI builder پشتیبانی نمی کند. می توانید برای مشاهده متن خطای مربوطه (عدم امکان استفاده از GestureOverlayView در layout builder) به آدرس <http://code.google.com/p/android/issues/detail?id=1368> مراجعه کنید. از این رو توسعه دهنده ناگذیر باید یک layout بدون gesture view ساخته و بعد این view را از طریق کدنویسی به GestureOverlayView اضافه کند.

فایل layout\_main.xml را مانند زیر ایجاد کنید.

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

```
<TextView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:text="@string/hello"/>
</LinearLayout>
```

کلاس activity زیر را پیاده سازی نمایید.

```
package de.vogella.android.gestures;
import java.util.ArrayList;
import android.app.Activity;
import android.gesture.Gesture;
import android.gesture.GestureLibraries;
import android.gesture.GestureLibrary;
import android.gesture.GestureOverlayView;
import android.gesture.GestureOverlayView.OnGesturePerformedListener;
import android.gesture.Prediction;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;
public class GestureTest extends Activity implements OnGesturePerformedListener {
    private GestureLibrary gestureLib;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        GestureOverlayView gestureOverlayView = new GestureOverlayView(this);
        View inflate = getLayoutInflater().inflate(R.layout.main, null);
        gestureOverlayView.addView(inflate);
        gestureOverlayView.addOnGesturePerformedListener(this);
        gestureLib = GestureLibraries.fromRawResource(this, R.raw.gestures);
        if (!gestureLib.load()) {
            finish();
        }
        setContentView(gestureOverlayView);
    }
    @Override
    public void onGesturePerformed(GestureOverlayView overlay, Gesture gesture) {
        ArrayList<Prediction> predictions = gestureLib.recognize(gesture);
        for (Prediction prediction : predictions) {
            if (prediction.score > 1.0) {
                Toast.makeText(this, prediction.name, Toast.LENGTH_SHORT)
                    .show();
            }
        }
    }
}
```

اپلیکیشن را اجرا کنید. با توجه به منطق برنامه و کدی که برای اپلیکیشن خود نوشتشد، باید بتوانید gesture های مختلفی ترسیم کرده و خروجی را در لحظه مشاهده نمایید. در اپلیشکن جاری تنها صرفا یک پیغام Toast نمایش داده می شود ولی شما می توانید هر عملیاتی که دوست دارید تعریف نمایید.





902

آدرس آموزشگاه : تهران - خیابان شریعتی - بالاتر از خیابان ملک - جنب بانک صادرات - پلاک 651 طبقه دوم - واحد 7

88146323 - 88446780 - 88146330

## بخش اول :

# آموزش استفاده از Support Library ها

آموزش حاضر نحوه‌ی استفاده از support library ها و همچنین دلایلی که چرا گاهی بهتر است از بکار بردن آن‌ها در پروژه پرهیز کرد را شرح می‌دهد.

## استفاده از Google های Support library

### شرح مفهوم Support Library

این امکان را به توسعه دهنده می‌دهد تا قابلیت‌های نوین و جدید اضافه شده به Android SDK یا مجموعه ابزار ساخت و توسعه‌ی اپلیکیشن‌های اندروید را به همراه برنامه قرار دهد تا بدین وسیله برنامه قادر باشد از این امکانات در ویرایش‌های قدیمی اندروید نیز بهره بگیرد. به عبارت دیگر support library ها به منظور افزودن امکانات و قابلیت‌های ورژن‌های جدیدتر توابع و کتابخانه‌های اندروید به ویرایش‌های قدیمی‌تر تعییه شدند.

پکیج Android Support Library در واقع مجموعه‌ای از code library ها (کتابخانه‌ای از کدهای آماده) هستند که نسخه‌های سازگار با ورژن‌های قدیمی‌تر از امکانات جدید کتابخانه‌های اندروید را به توابع کتابخانه‌ای و چارچوب نرم افزاری اندروید اضافه می‌کنند. علاوه بر آن ویژگی‌ها و widget هایی را برای استفاده در ورژن‌های قدیمی کتابخانه‌های اندروید فراهم می‌آورند که فقط و فقط از طریق API ها و توابع کتابخانه‌ای قابل دسترسی یا استفاده در پروژه هستند.

لازم به توضیح است که هر support library تنها برای ورژن (قدیمی) معینی از کتابخانه‌های اندروید (API level) قابل استفاده بوده و سازگار می‌باشد.

در کل دو نوع support library وجود دارد. اولین نوع قابلیت‌های جدید برای ورژن‌های قدیمی تر چارچوب نرم افزاری فراهم می‌کند (framework functionality) جدید برای ویرایش‌های قدیمی‌تر کتابخانه‌های اندروید یا API release ارائه می‌دهد. نوع دوم support library، قابلیت‌های جدید

برای تمامی دستگاه های مبتنی بر سیستم عامل اندروید فراهم می کند. حال این قابلیت یا هنوز به محیط (platform) اندروید اضافه نشده یا هیچگاه جزوی از آن نخواهد بود زیرا شرکت گوگل سعی بر این دارد که همواره امکانات و قابلیت های جدیدتر به آن اضافه کند و کتابخانه ی مزبور را سریع تر از توابع کتابخانه ای محیط اندروید (Android platform API) ارتقا دهد.

## DADEH Support library نصب

به شما این امکان را می دهد تا از قابلیت های بیشتری در اپلیکیشن های اندرویدی خود استفاده کنید. حال این قابلیت ها ممکن است توسط ورژن های جدید کتابخانه های اندروید (API level) در اختیار توسعه دهنده قرار گیرد. علاوه بر آن امکانات دیگری نیز می تواند فراهم نماید که به صورت مجزا (unbundled) از چارچوب نرم افزاری اندروید، همچون ویجت RecyclerView که به منظور پیاده سازی هر چه بهینه تر لیست تعییه شده، ارائه می شود.

اندروید ورژن های متعددی از support library ها ارائه می دهد که هر یک، ویژه ی نسخه ی خاصی از کتابخانه های اندروید (API level) قابل استفاده می باشد. به طور مثال، support library v7 برای آن دسته از دستگاه های اندروید که API level آن ها 7 باشد، قابل استفاده می باشد. نسخه های جدیدتر از support library برای عملکرد صحیح به نسخه های پایین تر نیز احتیاج دارند. support library v7 به v4 احتیاج دارد.

## Support library هایی که توسط گوگل ارائه می شوند

در زمان تنظیم مقاله ی حاضر، کامپوننت های متعددی توسط شرکت گوگل تعییه شده که در زیر به آن ها اشاره می کنیم:

Support Library	شرح	Gradle dependency کتابخانه و نیازمندی های
v4 Support Library	ویژه ی ویرایش API ( Android 1.6 level 4 ) و بالاتر تعییه شده است.	این کتابخانه در آدرس android-sdk/extras/android/support/v7/appcompat/directory موجود می باشد.

Support Library	شرح	Gradle dependency کتابخانه و نیازمندی های
	دربردارنده‌ی قابلیت‌های فراوانی نظریه امکان پشتیبانی از ها و fragment فریم‌ورک Loader می‌باشد.	
v7 Support Libraries		مجموعه‌ای از کتابخانه‌ها که به Android 2.1 (کتابخانه‌های اندروید ورژن 7) و بالاتر برای عملکرد صحیح احتیاج دارند. علاوه بر آن لازم است v4 Support library در دسترس باشد.
v7 appcompat library	امکان پشتیبانی از و پیاده Action Bar سازی آن در بستر اپلیکیشن را فراهم می‌آورد. علاوه بر به توسعه دهنده امکان می‌دهد برای طراحی لایه‌ی UI اپلیکیشن از Material design استفاده کند.	در آدرس android-sdk/extras/android/support/v7/appcompat/ موجود می‌باشد.
v7 cardview library	با افزودن این نیازمندی به پروژه قادر خواهید بود که CardView ویجت را در اپلیکیشن خود پیاده سازی کنید.	android-sdk/extras/android/support/v7/cardview/

Support Library	شرح	Gradle dependency کتابخانه و نیازمندی های
v7 recyclerview library	امکان استفاده از RecyclerView را پروژه فراهم می آورد.	compile "com.android.support:recyclerview-v7:24.0.0"
Design library to support material design	امکان استفاده از material design در طراحی رابط کاربری اپلیکیشن را فراهم می آورد.	compile 'com.android.support:design:24.0.0'

حذف support library از پروژه

چرا باید support library را حذف کرد؟

همواره در حال تغییر هستند و از این جهت تنظیم یک مقاله‌ی پایدار (با مطالبی که طولانی مدت بروز و مفید باشند) در شرح نحوه‌ی استفاده از این کتابخانه‌ها کاری بسیار دشوار است. علاوه بر آن الگو و قالب‌های آماده‌ی ساخت پروژه‌های اندرویدی در محیط کاری Android studio نیز همانند کتابخانه‌ی نام بردۀ به سرعت بروز رسانی و ویرایش می‌شوند. این امر تست ویژگی جدید یا API‌های استاندارد را به مراتب دشوارتر می‌سازد.

## تبديل پروژه به یک پروژه‌ی استاندارد اندروید

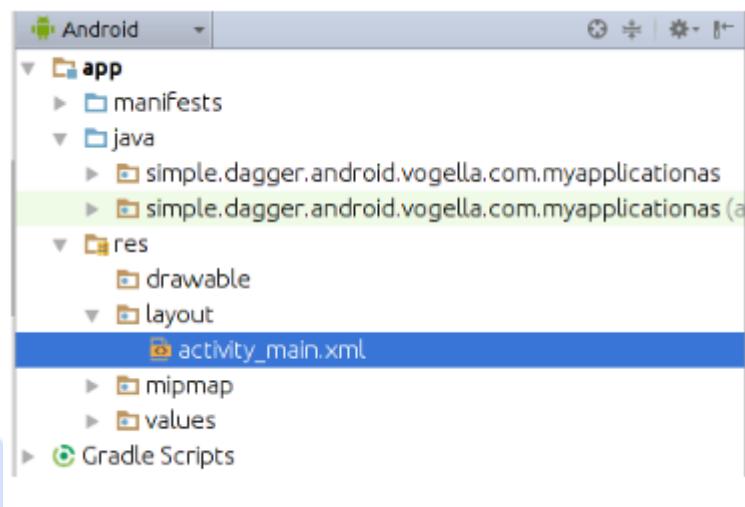
کتابخانه‌ها و نیازمندی‌های build dependency (build dependency آسیبی به پروژه‌ی شما نمی‌رساند، اما توصیه می‌شود ارجاعات یا اشاره‌گر (reference) به resource را از کد و فایل‌های compatibility library / library خود حذف نمایید).

برای این منظور، کد کلاس activity را طوری ویرایش کنید که منحصر از کلاس پایه‌ی Activity ارث بری داشته باشد (نه از کلاس‌های AppCompatActivity نظیر). همچنین لازم

است کد را بینه و تا حد ممکن مختصر تنظیم نمایید. در زیر نمونه ای پیاده سازی یک کلاس است که را مشاهده می کنید.

```
package com.vogella.android.myapplication;
import android.app.Activity;
import android.os.Bundle;
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

همچنین سبب می شود تنها المان های استاندارد از فایل layout متناظر استفاده کنند. در زیر نمونه ای از یک فایل layout استاندارد را مشاهده می کنید.



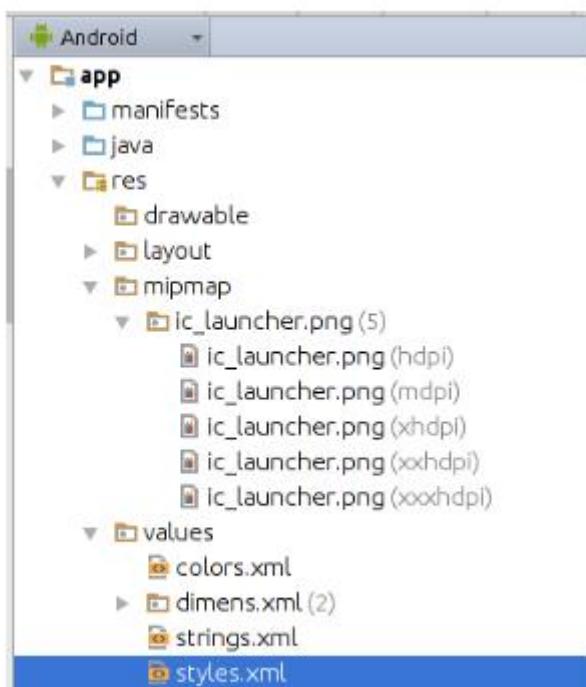
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
```

```

tools:context=".MainActivity">
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World!" />
</RelativeLayout>

```

به جای `theme` هایی که با ورژن های قدیمی اندروید نیز سازگاری دارند، حتماً از `compatibility theme` استفاده کنید. برای این منظور محتوا پوششی `material theme` را در `app/res/values/styles.xml` (یک `style` جدید) بروزرسانی کنید.



```

<resources>
    <!-- Base application theme. -->
    <style name="AppTheme" parent="android:Theme.Material.Light.DarkActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>
</resources>

```

## بخش دوم :

# نوشتن کتابخانه برای اپلیکیشن های اندرویدی

آموزش حاضر به شرح ایجاد و استفاده از پروژه های کتابخانه ای (library project) در اندروید می پردازد.

## پروژه های کتابخانه ای اندروید و کتابخانه های جاوا

### استفاده از کتابخانه های اندروید و جاوا

یک پروژه ای اندرویدی به راحتی می تواند از کدهای موجود در فایل های JAR (فایل های فشرده حاوی چندین کلاس جاوا به همراه metadata و منابع مورد نیاز که به کتابخانه های جاوایی معروف هستند) استفاده کند.

جدا از فایل های JAR، اندروید از یک فرمت توزیع نرم افزار (binary distribution) ویژه به نام Android ARchive(AAR) (کتابخانه هایی که جدا از کلاس های جاوا دربردارنده ای انواع فایل ها و منابع مبتنی بر XML می باشند) استفاده می کند. در واقع یک پروژه ای کتابخانه ای اندروید است با روش توزیع نرم افزار binary distribution (به صورت کامپایل شده) ارائه می شود.

یک فایل AAR تقریباً شبیه به یک فایل JAR است اما علاوه بر کلاس های نوشته شده با جاوا، حامل سایر منابع و محتوا و نیز byte-code کامپایل شده می باشد. فایل AAR را می توان در فرایند build و کامپایل اپلیکیشن درست مانند فایل JAR بکار برد.

توسعه دهنده می تواند مازول های کتابخانه ای ایجاد کرده و از آن ها به عنوان dependency در پروژه های اندرویدی استفاده کند. این مازول ها به شما امکان می دهد تا کد برنامه (source code) و منابعی همچون عکس و متن را ذخیره کرده و بین سایر پروژه های اندرویدی به اشتراک گذاشته و استفاده کنید.

## نحوه استفاده از فایل JAR در اپلیکیشن اندرویدی

به منظور استفاده از یک فایل JAR (فایل کتابخانه ای جاوا) در پروژه، کافی است آن فایل را داخل پوشه `libs` در اپلیکیشن خود جایگذاری کنید. فایل های jar داخل این پوشه از طریق فایل `build.gradle` پیش فرض، در `compile classpath` قرار داده می شوند.

## محدودیت هایی در استفاده از کتابخانه های java

در صورت نیاز به استفاده از کتابخانه ها لازم است این نکته را به خاطر داشته باشید که این کتابخانه ها تنها باید از API ها و توابع متعارف و شناخته شده در اندروید استفاده کنند. به طور مثال، کتابخانه های اندروید کتابخانه های لازم برای طراحی رابط کاربری اپلیکیشن همچون `java.awt` و `javax.swing` را ندارند. کتابخانه ای که کلاس های موجود در این کتابخانه ها را فراخوانی کند، قابل استفاده در اندروید نخواهد بود.

پروژه های کتابخانه ای قابل کامپایل و تبدیل به اپلیکیشن های کامل اندروید نبوده و به صورت مستقل از پروژه ای بزرگتر که از کلاس ها و توابع آن در خود استفاده کند، قابل اجرا نخواهند بود.

## ماژول های اختصاصی کتابخانه ای اندروید (library modules)

### ماژول های کتابخانه ای (library module)

زمانی که یک پروژه (application project) از ماژول کتابخانه (Android library module) یا `build` (library module) استفاده می کند، ابزار ساخت و توسعه `ADT` این ماژول را از اپلیکیشن های اندروید / `apk` کد و منابع لازم را از پروژه ای کتابخانه ای در خروجی کامپایل و `build` اپلیکیشن های اندروید می گنجاند. این بدین معنی است که کامپوننت ها، کد و منابع پروژه کتابخانه ای شده ای پروژه می شوند. آنچه کامپوننت ها، کد و منابع پروژه کتابخانه ای (library project) کاملا ترجمه شده و همگی در قالب فایل `apk` اپلیکیشن کامپایل شده پکیج بندی می شوند.

از اینرو می توان به library module به چشم یک مصنوع زمان کامپایل نگاه کرد. مازول کتابخانه ای می تواند شامل کلاس های جوا، کامپوننت های نرم افزاری اندروید و منابع باشد. تنها نوع فایلی که در این کتابخانه ها جایی ندارد، asset ها هستند.

جهت ایجاد یک پروژه کتابخانه، در ویزارد ساخت پروژه های اندرویدی (Android project) گزینه **Mark this project as library** (generation wizard) را انتخاب کنید.

یک پروژه کتابخانه ای بایستی تمامی کامپوننت های خود نظیر activity ها، سرویس و غیره ... را در لایه XML، داخل فایل تنظیمات AndroidManifest.xml اعلام کند. اپلیکیشنی که از این کتابخانه استفاده می کند نیز بایستی تمامی کامپوننت های نرم افزاری مورد استفاده را داخل فایل تنظیمات اپلیکیشن مربوطه اعلام نماید.

### اولویت در انتخاب منابع (conflicting resources)

ابزار ساخت و توسعه اپلیکیشن های اندرویدی (Android development tool) منابع پروژه کتابخانه را با منابع پروژه اپلیکیشن ادغام می کند. حال چنانچه ID یا شناسه ای منابع پروژه چندین بار به صورت تکراری اعلام شده باشد، ADT تلاش می کند آن منبعی را از اپلیکیشن یا کتابخانه گزینش نماید که بالاترین اولویت را داشته و منابع اضافی را دور می ریزد.

### ایجاد مازول های کتابخانه ای اختصاصی در اندروید (Android library module)

استفاده از پروژه های کتابخانه ای (library project) به توسعه دهنده کمک می کند تا کد اپلیکیشن را به صورت بهینه سازمان دهی کند. جهت ایجاد یک مازول کتابخانه ای جدید در محیط برنامه نویسی Android Studio، بر روی المان File از نوار بالایی محیط کلیک کرده و گزینه **New Module** را باز نمایید و سپس **Android Library** را انتخاب کنید.

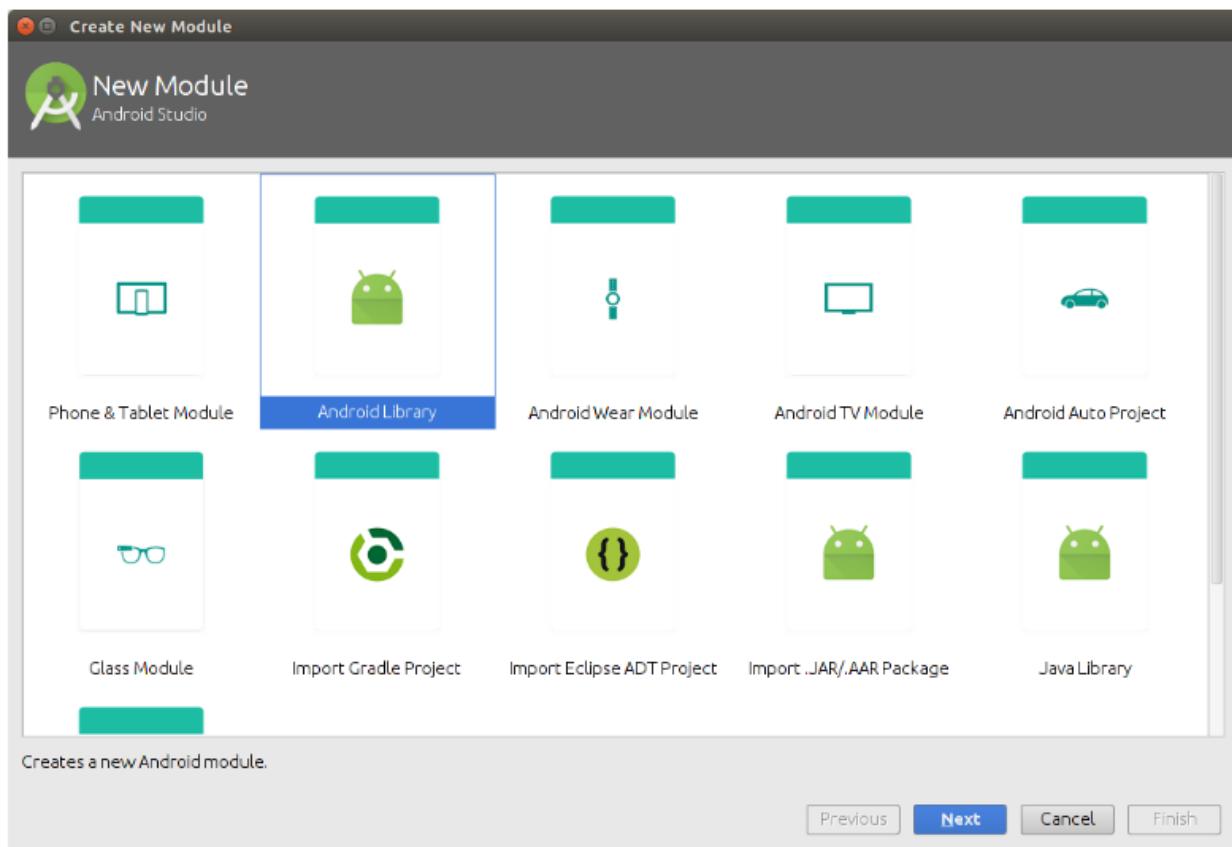
### تمرین: ساخت یک library module (کتابخانه ای اندروید)

برای تمرین جاری لازم است پروژه ای که در مباحث قبلی ایجاد کرده و اسم پکیج (top level package) آن com.example.android.rssfeed باشد را در اختیار داشته باشید (می توانید به مقاله ای آموزش fragment ها مراجعه نمایید). پروژه ای کتابخانه ای که ایجاد می کنیم علاوه بر کلاس های ساخته شده از جداول دیتابیس، یک تابع جهت بازیابی تعداد نمونه ها (instances) در برخواهد داشت. کتابخانه ای مورد نظر اجازه دسترسی به داده ها و اطلاعات (RSS) را می دهد. فایل RSS نیز صرفا یک فایل XML است که برای انتشار و بارگذاری اخبار و نوشته ها و پست های وبلاگ مورد استفاده قرار می گیرد.

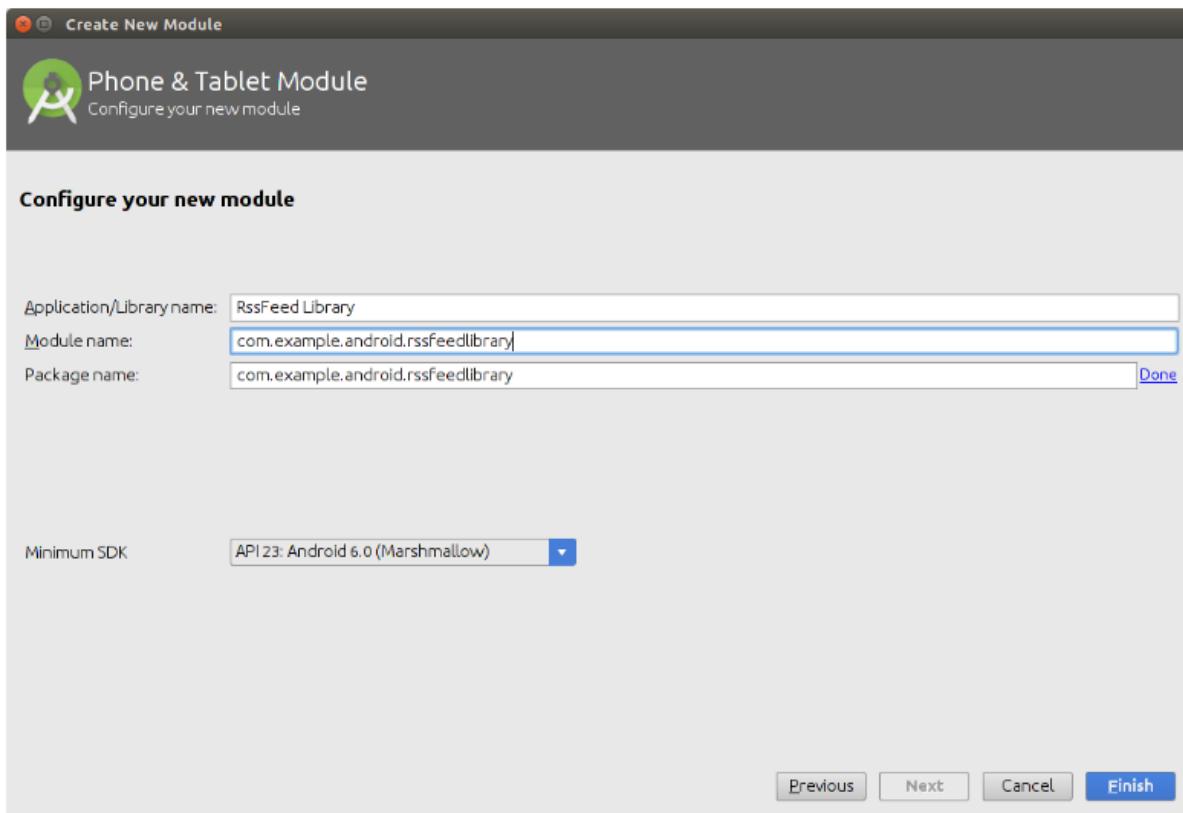
### ایجاد یک کتابخانه جدید (library module)

در محیط برنامه نویسی Android Studio هر کتابخانه خود یک ماژول محسوب می شود. برای ایجاد یک ماژول یا کتابخانه جدید در محیط کاری نام برد، این مسیر را طی نمایید:

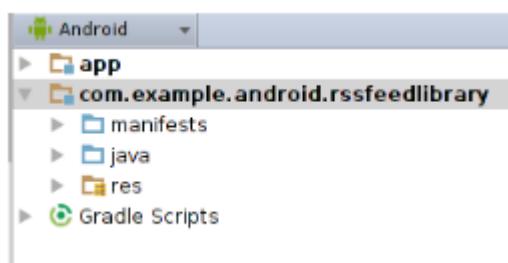
File > New Module > Android Library



به عنوان اسم ماژول، com.example.android.rssfeedlibrary و به عنوان اسم کتابخانه Rssfeed Library را وارد نمایید.



زمانی که از شما درخواست می شود که قالب آماده یا template دلخواه را انتخاب نمایید، صریحاً اعلان کنید که نمی خواهید activity ساخته شود. در نتیجه محیط کاری Android Studio یک ماژول دیگر برای شما به نمایش می گذارد.



## ایجاد کلاس (data model) model

یک کلاس به نام `RssItem` جهت ذخیره ی داده های مربوط به محتوای RSS ایجاد نمایید.

حال توابع `getter` و `setter`, تابع سازنده `toString()` و یک متد `(constructor)` در کلاس اعلان نمایید. خروجی بایستی ظاهری مشابه زیر داشته باشد:

```
package com.example.android.rssfeedlibrary;
public class RssItem {
    private String pubDate;
    private String description;
    private String link;
    private String title;
    public RssItem() {
    }
    public RssItem(String title, String link) {
        this.title = title;
        this.link = link;
    }
    public String getPubDate() {
        return pubDate;
    }
    public void setPubDate(String pubDate) {
        this.pubDate = pubDate;
    }
    public String getDescription() {
        return description;
    }
    public void setDescription(String description) {
        this.description = description;
    }
    public String getLink() {
        return link;
    }
    public void setLink(String link) {
        this.link = link;
    }
    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
        this.title = title;
    }
    @Override
    public String toString() {
        return "RssItem [title=" + title + "]";
    }
}
```

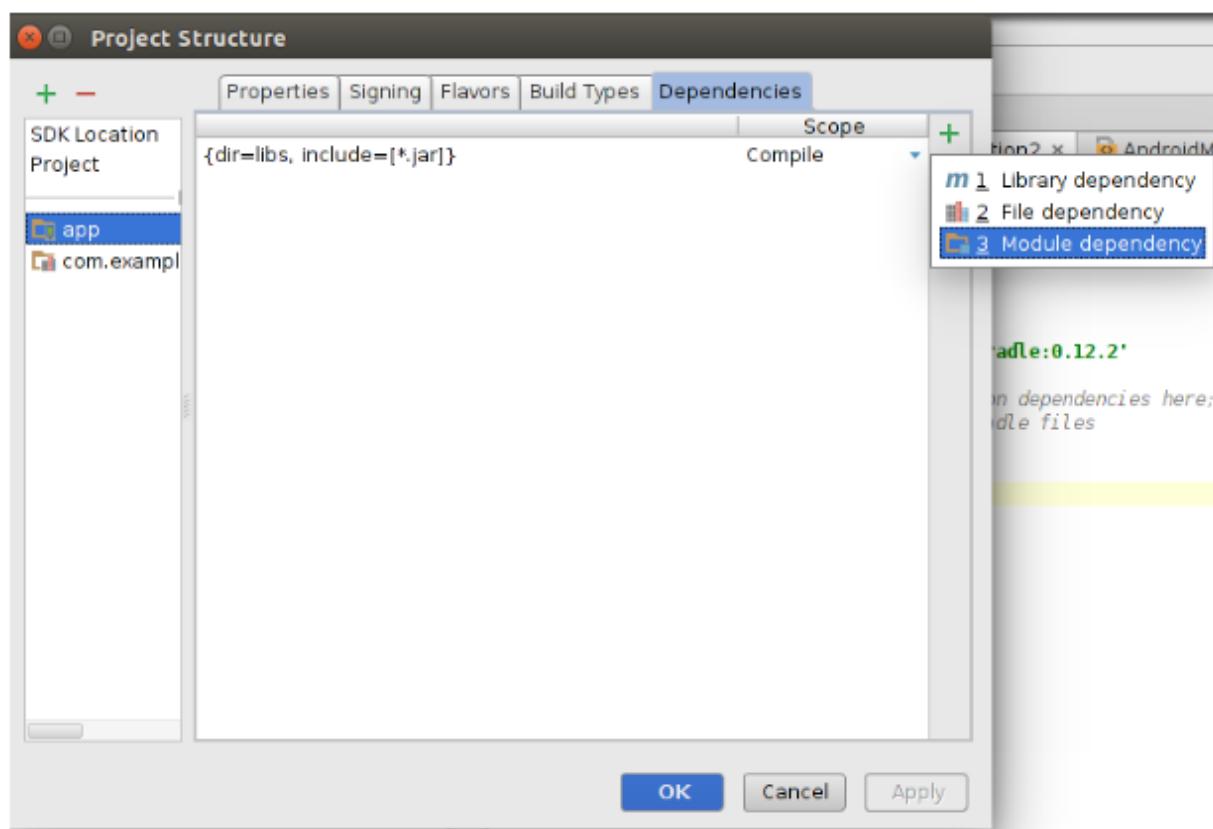
## ایجاد کلاس های مورد نیاز (instances)

اکنون یک کلاس جدید به نام RssFeedProvider با یک متدهای static که لیستی از آبجکت های RssItem را در خروجی برمی گرداند، تعریف نمایید. این متدهای حال حاضر تنها داده های آزمایشی (test data) را برمی گرداند.

```
package com.vogella.android.rssfeedlibrary;
import java.util.ArrayList;
import java.util.List;
import java.util.Random;
public class RssFeedProvider {
    public static List<RssItem> parse(String rssFeed) {
        List<RssItem> list = new ArrayList<>();
        Random r = new Random();
        // random number of item but at least 5
        Integer number = r.nextInt(10) + 5;
        for (int i = 0; i < number; i++) {
            // create sample data
            String s = String.valueOf(r.nextInt(1000));
            RssItem item = new RssItem("Summary " + s, "Description " + s);
            list.add(item);
        }
        return list;
    }
}
```

## اضافه کردن کتابخانه به عنوان dependency به پروژه

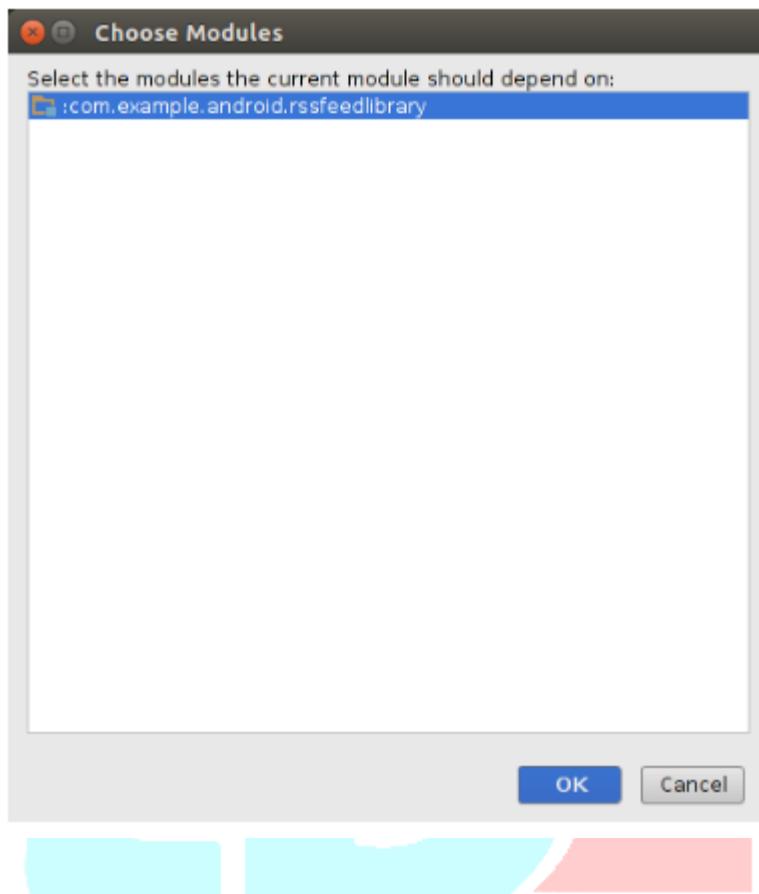
به منظور استفاده از کتابخانه، کافی است آن را به عنوان dependency در بستر پروژه خود وارد کنید. برای این منظور مسیر رو به رو را طی نمایید: File ▶ Project Structure. پوشه app را انتخاب کنید. حال تب Dependencies را باز نموده و پس از کلیک بر روی علامت +، المان module dependencies را از فهرست انتخاب نمایید.



916

آدرس آموزشگاه: تهران - خیابان شریعتی - بالاتر از خیابان ملک - جنب بانک صادرات - پلاک 651 طبقه دوم - واحد 7

88146323 - 88446780 - 88146330

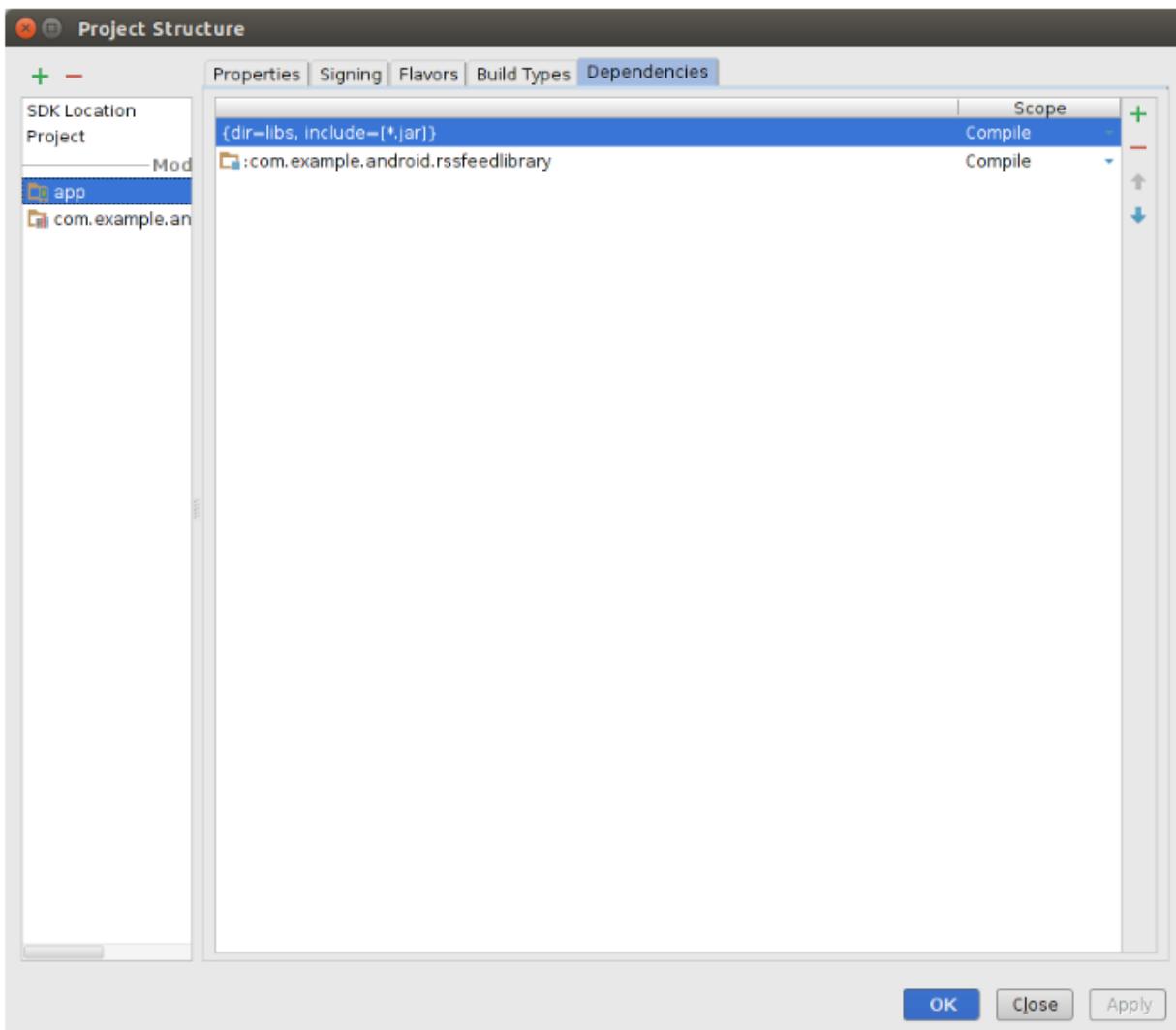


آموزشگاه تخلیکر داده ها

917

آدرس آموزشگاه : تهران - خیابان شریعتی - بالاتر از خیابان ملک - جنب بانک صادرات - پلاک 651 طبقه دوم - واحد 7

88146323 - 88446780 - 88146330



## استفاده از پروژه کتابخانه جهت بروز رسانی داده های details fragment

متد updateDetail را جهت استفاده از کامپوننت نرم افزاری RssFeedProvider، در بدنه ی کلاس myListFragment بروز رسانی نمایید. کدهای موجود در آن صرفا آزمایشی (test code) هستند.

```
public class myListFragment extends Fragment {
    // everything as before
    // triggers update of the details fragment
    public void updateDetail(String uri) {
        List<RssItem> list = RssFeedProvider
            .parse("http://www.vogella.com/article.rss");
        String itemListAsString = list.toString();
    }
}
```

(1) متدهای آپدیت

```

        listener.onRssItemSelected(itemListAsString);
    }
}

```

تست اپلیکیشن و بررسی عملکرد صحیح آن  
 اپلیکیشن خود را اجرا کرده و مطمئن شوید که خروجی تابع ()  
 (ToString) مقدار از detail fragment RssItems نمایش داده می شود.

**نکته:** لازم به ذکر است که لیست، با توجه به منطق و کدی که در حال حاضر برای اپلیکیشن تنظیم شده، صرفا به صورت تصادفی ایجاد می شود.



### بخش سوم :

919

آدرس آموزشگاه : تهران - خیابان شریعتی - بالاتر از خیابان ملک - جنب بانک صادرات - پلاک 651 طبقه دوم - واحد 7

88146323 - 88446780 - 88146330

# معرفی کتابخانه های کارا و پرکاربرد اندروید / Android

## پروژه های کتابخانه ای اندروید و Library project

مقاله‌ی حاضر تمامی کتابخانه‌های پرکاربرد را به همراه code repository های (انباری از کدهای آماده) کارا و محبوب اندروید معرفی می‌نماید.

### پروژه های کارامد و پرکاربرد اندروید

ویژگی متن باز و رایگان بودن (open source) اندروید سبب شده تا سایر توسعه دهندگان بتوانند به راحتی کتابخانه‌ها و ماثوله‌های کارا و اضافی بر سازمان (جدا از خود چارچوب نرم افزاری اندروید) برای آن طراحی کرده و در اختیار همگان قرار دهند. مبحث جاری کتابخانه‌های open source قدرتمند و نمونه پروژه‌های مفید که برنامه نویس را در توسعه‌ی هر چه بهینه‌تر پروژه‌ی خود یاری می‌کند، فهرست می‌نماید.

### لیست کتابخانه‌های پرکاربرد اندروید

- فریم ورک بارگذاری فایل‌های تصویری برای اندروید (Glide Image loading framework) - <https://github.com/bumptech/glide>
- فریم ورک - <http://www.vogella.com/tutorials/AndroidButterknife/article.html>
- نرم افزاری تزریق view و المان‌های رابط کاربری (view injection framework) - <http://www.vogella.com/tutorials/AndroidButterknife/article.html>
- یک event bus - Otto - <https://github.com/square/otto> که از جمله قابلیت‌های آن آسان سازی تعامل و تبادل داده بین کامپوننت‌های مختلف نرم افزاری اندروید می‌باشد.
- یک کلاینت type-safe - Retrofit - <http://square.github.io/retrifit/> و نوع ایمن REST اندروید و جاوا
- موتور ترسیم chart و AChartEngine - <http://code.google.com/p/achartengine/> نمودار
- جهت تولید دیتابیس برای اپلیکیشن‌های اندرویدی / ORM mapper - <http://ormlite.com/>
- Markus Junginger GreenDAO - <http://greendao-orm.com/> تعبیه شده توسط ORM Mapper

- Dagger2 - فریم ورک تزریق نیازمندی و dependency injection برای پروژه های جاوا و اندروید
- اسکریپت نویسی فراهم می کند تا بر روی محیط اندروید اجرا شوند - <http://code.google.com/p/android-scripting/> - این زمینه را برای زبان های

## و انباره نمونه کدها و پروژه ها Code repository

بهترین منبع برای (آشنایی با نحوه استفاده از توابع کتابخانه ای اندروید و API ها) نمونه کدهای اندروید API demos می باشد.

Roman Nurik، یک توسعه دهنده در شرکت گوگل، نمونه کدهای خود را تحت آدرس زیر در اختیار علاقه مندان قرار می دهد: - <http://code.google.com/p/romannurik-code/>

Roman Nurik's examples

Mark Murthy، مدرس دوره های برنامه نویسی اندروید، توسعه دهنده و نویسنده کتاب های

برنامه نویسی، پروژه ها و نمونه کدهای پیشرفته ای خود را تحت آدرس

<https://github.com/commonsguy/cw-advandroid> - Mark Murthy's advanced examples

در اختیار توسعه دهندگان قرار می دهد.

آموزشگاه تخلیکر داده ها

## بخش چهارم :

# آموزش استفاده از کتابخانه `Otto event bus` در پروژه های اندرویدی

مبخت حاضر به شرح نحوه ای استفاده از `Otto event bus` در اپلیکیشن های اندروید می پردازد.

`Otto` یک پروژه ای متن باز و رایگان (*open source*) است که به کامپوننت ها اجازه می دهد تا مکانیزم `event bus` (واسط و مترجم پیغام بین فرستنده و گیرنده) را پیاده سازی کرده تا کامپوننت های بتوانند به `event` ها گوش داده (`subscribe`) و به آن ها پیغام ارسال کنند (`publish`).

امکان برقراری ارتباط به سبک `publish-subscribe` را بین کامپوننت های نرم افزاری مختلف فراهم می سازد. بدین معنی که کامپوننت ها بدون اینکه لازم باشد به طور صریح به یکدیگر معرفی شده و اطلاعی از وجود هم داشته باشند، ارتباط برقرار کنند.

`Otto` یک شاخه از کتابخانه `Guava event bus` ساخت `Google` بوده و طوری بازطراحی شده که تا حد ممکن با `Android` سازگار باشد.

## نصب کتابخانه

استفاده از این کتابخانه در جاوا یا اندروید بسیار ساده است، کافی است فایل `JAR` و تنها نیازمندی آن را از `http://square.github.io/otto` دانلود کرده و سپس آن را به متغیر `classpath` اپلیکیشن اضافه نمایید.

در صورت استفاده از `Gradle` یا `Maven` به عنوان سیستم کامپایل و `build` پروژه، شما می توانید به راحتی کتابخانه مورد نیاز (`dependency`) را به شناسه `گروه (Group ID)` (`com.squareup.otto`) و `ورژن 1.3.5` (`artifact`) اضافه نمایید.

## چه زمانی باید از Otto استفاده کرد؟

Otto ابزار خوبی برای مهیا کردن امکان ارتباط بین fragment های زیرمجموعه‌ی آن و activity های زیرمجموعه‌ی آن و service می‌باشد.

### چگونگی تنظیم Otto

**توجه:** آموزش حاضر فرض را بر این می‌گذارد که توسعه دهنده کتابخانه‌ی Otto را در یک اپلیکیشن اندرویدی مورد استفاده قرار می‌دهد، اگرچه امکان استفاده از آن در بستر پروژه‌های جاوایی نیز وجود دارد.

جهت استفاده از Otto، لازم است یک نمونه‌ی واحد (singleton instance) از کلاس Bus ایجاد کرده و سپس برای کامپوننت‌های نرم افزاری اندروید امکان دسترسی به آن را فراهم نمایید. عملیات مذبور در آبجکت Application پروژه‌ی اندرویدی شما انجام می‌شود.

```
public static Bus bus = new Bus(ThreadEnforcer.MAIN);
```

در مثال جاری، همان طور که مشاهده می‌کنید، مقدار ThreadEnforcer.MAIN به عنوان پارامتر به تابع سازنده‌ی کلاس Bus ارسال شده است. به واسطه‌ی این پارامتر، Otto، به سیستم اعلان می‌کند که event‌ها منحصر باشد از thread اصلی ارسال شوند. اگر مهم نیست که رخداد از کدام thread ارسال می‌شود، پارامتر ThreadEnforcer.ANY به عنوان آرگومان به تابع سازنده‌ی Bus پاس داده می‌شود.

### نحوه‌ی register یا گوش دادن به event‌ها و unregister آن‌ها

جهت گوش دادن به رخدادها (event registration)، لازم است دستور @Subscribe را در بالای متدهای با سطح دسترسی public تعریف شده و فقط یک پارامتر ورودی دارد، درج نمایید. پارامتر ارسالی به متدهای در واقع event key محسوب می‌شود، بدین معنی که چنانچه چنین نوع داده‌ای (data type) به واسطه‌ی event bus (واسطه و مدیریت کننده‌ی رخدادها) ارسال شود، در آن صورت متدهای مرتبط صدا زده می‌شود.

متدهایی که رخداد را دریافت می کنند) بایستی خود را به وسیله `i` تابع از کلاس Bus ثبت کرده و به رخداد مربوطه گوش فرا دهند.

```
// subscribe for string messages
@Subscribe
public void getMessage(String s) {
    Toast.makeText(this, s, Toast.LENGTH_LONG).show();
}

//subscribe for TestData messages
@Subscribe
public void getMessage(TestData data) {
    Toast.makeText(getActivity(), data.message, Toast.LENGTH_LONG).show();
}

//requires a registration e.g. in the onCreate method
bus.register(this);
```

جهت unregister کردن و قطع فرایند گوش دادن به event ها، کافی است متدهای `unregister()` را فراخوانی نمایید.

## نحوه ارسال event ها

برای ارسال رخداد نیازی به `register` در event bus نیست. کافی است متدهای `post()` از کلاس Bus را فراخوانی نمایید (متدهای `post` را بر روی آبجکت ساخته شده از کلاس Bus صدا بزنید).

```
// post a string object
bus.post("Hello");
// example data to post
public class TestData {
    public String message;
}
// post this data
bus.post(new TestData().message="Hello from the activity");
```

چگونه کامپوننت های نرم افزاری جدید می توانند آخرین event را دریافت کنند؟

در صورتی که کامپوننت های جدید، نظری یک fragment که به صورت dynamic و در زمان اجرا ایجاد شده، داده های مربوط به event را در طول فرایند ایجاد دریافت کنند، کامپوننت های نرم افزاری می توانند از طریق دستور `@Produce` خود را به عنوان producer یا تولید کننده معرفی کنند. رخداد `register` معرفی شده است.

کامپوننت هایی که می خواهند به event گوش داده و قرار است داده های مربوط به رخداد را دریافت کنند) بایستی متدهای register را از کلاس Bus فراخوانی نمایند.

```
@Produce
public String produceEvent() {
    return "Starting up";
}
```

## Otto event bus تمرین: استفاده ای کاربردی از کتابخانه ای

مثال حاضر کتابخانه ای Otto را به صورت کاربردی در یک پروژه ای اندرویدی مورد استفاده قرار می دهد. هرچند این کتابخانه در اپلیکیشن های جاوا نیز قابل فراخوانی و استفاده می باشد.

یک پروژه ای اندرویدی به نام com.vogella.java.library.otto، بر اساس قالب آماده ای Empty Activity with Fragment (template) ایجاد نمایید.

چنانچه برای برنامه نویسی از محیط کاری Eclipse استفاده می کنید، در آن صورت فایل Otto JAR را دانلود کرده و سپس آن را به build path پروژه ای خود اضافه نمایید.

در صورت استفاده از Gradle، کافی است کتابخانه (dependency) مورد نیاز را به فایل build.gradle اضافه نمایید.

```
apply plugin: 'com.android.application'
android {
    compileSdkVersion 22
    buildToolsVersion "21.1.2"
    defaultConfig {
        applicationId "com.example.android.rssreader"
        minSdkVersion 22
        targetSdkVersion 22
        versionCode 1
        versionName "1.0"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}
dependencies {
    compile fileTree(include: ['*.jar'], dir: 'libs')
    compile project(':com.example.android.rssfeedlibrary')
```

```

    compile 'com.squareup:otto:1.3.5'
    compile 'org.achartengine:achartengine:1.2.0'
}

```

محتوای فایل تنظیم ظاهر اپلیکیشن، activity\_main.xml layout، بایستی ظاهری مشابه زیر داشته باشد.

```

<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/container"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.vogella.android.otto.MainActivity"
    tools:ignore="MergeRootFrame" />

```

فایل تنظیم کننده ی ظاهر fragment\_main.xml، fragment，بایستی محتوای مشابه زیر داشته باشد.

```

<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"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.vogella.android.otto.MainActivity$PlaceholderFragment" >
    <Button
        android:id="@+id/fragmentbutton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:text="Send event from fragment" />
</RelativeLayout>

```

پس از تعریف کردن fragment، آن را به صورت dynamic و داخل کلاس جاوایی به activity اضافه نمایید. سپس یک رخداد به ترتیب از activity به fragment و بالعکس ارسال کنید. در هر دو سناریو پس از دریافت رخداد یک پیغام Toast برای کاربر به نمایش بگذارید.

```

package com.vogella.android.otto;
import android.app.Activity;
import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;

```

```

import android.widget.Toast;
import com.squareup.otto.Bus;
import com.squareup.otto.Produce;
import com.squareup.otto.Subscribe;
import com.squareup.otto.ThreadEnforcer;
import library.java.vogella.com.otto.R;
public class MainActivity extends Activity {
    public static Bus bus;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        if (savedInstanceState == null) {
            getSupportFragmentManager().beginTransaction().add(R.id.container, new PlaceholderFragment()).commit();
        }
        bus = new Bus(ThreadEnforcer.MAIN);
        bus.register(this);
    }
    @Subscribe
    public void getMessage(String s) {
        Toast.makeText(this, s, Toast.LENGTH_LONG).show();
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();
        if (id == R.id.action_settings) {
            TestData t = new TestData();
            t.message="Hello from the activity";
            bus.post(t);
            return true;
        }
        return super.onOptionsItemSelected(item);
    }
    public class TestData {
        public String message;
    }
    /**
     * A placeholder fragment containing a simple view.
     */
    public static class PlaceholderFragment extends Fragment {
        public PlaceholderFragment() {
        }
        @Override
        public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
            View rootView = inflater.inflate(R.layout.fragment_main, container, false);
            View button = rootView.findViewById(R.id.fragmentbutton);
            button.setOnClickListener(new View.OnClickListener() {

```

```
@Override  
public void onClick(View v) {  
    bus.post("Hello from the Fragment");  
}  
});  
bus.register(this);  
return rootView;  
}  
  
@Subscribe  
public void getMessage(MainActivity.TestData data) {  
    Toast.makeText(getActivity(), data.message, Toast.LENGTH_LONG).show();  
}  
}  
  
@Produce  
public String produceEvent() {  
    return "Starting up";  
}  
}
```

**توجه:** مثال حاضر به طور عمده این گونه ساده طراحی شده است. در یک اپلیکیشن واقعی و تمام عیار اندروید، توسعه دهنده یک نمونه ی واحد (singleton) از کلاس Bus در آبجکت Application ایجاد کرده و فایل مجزای خود قرار می دهد.



929

آدرس آموزشگاه : تهران - خیابان شریعتی - بالاتر از خیابان ملک - جنب بانک صادرات - پلاک 651 طبقه دوم - واحد 7

88146323 - 88446780 - 88146330

## بخش اول

# آموزش مدیریت فایل های صوتی در اندروید / پیاده سازی توابع کتابخانه `Media API` جهت پخش و ضبط صدا

آموزش حاضر نحوه استفاده از توابع کتابخانه ای `media API` جهت پخش و ضبط صدا در محیط اندروید را شرح می دهد. پروژه ای این مبحث در محیط کاری Eclipse 3.7، با ویرایش 1.6 زبان `java` و بر اساس ورژن 4.0.3 سیستم عامل اندروید نوشته می شود.

### مدیریت صدا در محیط اندروید (`Android Sound`)

#### پخش صدا

اندروید دو مجموعه توابع کتابخانه ای یا API برای پخش صدا در اختیار توسعه دهندگان قرار می دهد. اولین مجموعه از این توابع کلاس `SoundPool` و دیگری کلاس `MediaPlayer` می باشد.

SoundPool را می توان برای کلیپ های کوچک صوتی مورد استفاده قرار داد. این کلاس قادر است صدایها را تکرار کرده و همچنین به طور همزمان چندین صدا را پخش کند. فایل های صوتی که توسط کلاس `SoundPool` پخش می شوند نباید از نظر حجم از مرز 1 مگابایت فراتر روند.

لازم به ذکر است که `SoundPool` فایل صوتی مربوطه را به صورت ناهمزمان (asynchronously) بارگذاری می کند. از ورژن 8 کتابخانه های اندروید (API 8) این امکان وجود دارد که از اتمام بارگذاری فایل و اطلاعات مورد نظر اطمینان حاصل نمایید. برای این منظور کافی است از `OnLoadCompleteListener` استفاده نمایید.

سیستم اندروید برای اهداف مختلف، از `audio stream` های (جريانی از بیت ها که حامل اطلاعات صوتی بوده و یک فایل صوتی را تشکیل می دهد) مختلف پشتیبانی می کند. می توان دکمه ای افزایش و کاهش میزان صدای دستگاه را طوری تنظیم کرد که برای مثال در حین

دریافت تماس، میزان صدای تماس گیرنده را کاهش یا افزایش دهد. جهت تنظیم دکمه برای کنترل media stream و اطلاعات صوتی (جريانی از بیت ها و اطلاعات که تشکیل دهنده فایل صوتی هستند)، کافی است audio type را در اپلیکیشن خود مقداری دهی و مشخص نمایید.

```
context.setVolumeControlStream(AudioManager.STREAM_MUSIC);
```

کلاس MediaPlayer گزینه‌ی بهتری جهت پخش موسیقی های طولانی تر و فایل‌های تصویری حجمی نظیر فیلم‌ها می‌باشد.

## کلاس MediaRecorder

با استفاده از کلاس MediaRecorder می‌توان به راحتی اطلاعات مربوط به صدا و تصویر را ضبط کرد. جهت استفاده از کلاس نام بردۀ لازم است دستگاه منبع (source device) و فرمت را مشخص نمایید.

## افزودن Media به Media library

این امکان برای شما وجود دارد که Media (فایل و داده‌های صوتی) جدید به media library (مجموعه فایل‌های صوتی) اندروید اضافه نمایید. برای نیل به این هدف کافی است یک آبجکت Intent ایجاد نموده و سپس از طریق آن media application (اپلیکیشن پخش و ضبط صدا) مستقر بر روی دستگاه، به آن اطلاع دهید که محتوای جدید در دسترس می‌باشد. کد زیر نحوه‌ی پیاده‌سازی آن را به نمایش می‌گذارد.

```
// add new file to your media library
ContentValues values = new ContentValues(4);
long current = System.currentTimeMillis();
values.put(MediaStore.Audio.Media.TITLE, "audio" + audiofile.getName());
values.put(MediaStore.Audio.Media.DATE_ADDED, (int) (current / 1000));
values.put(MediaStore.Audio.Media.MIME_TYPE, "audio/3gpp");
values.put(MediaStore.Audio.Media.DATA, audiofile.getAbsolutePath());
ContentResolver contentResolver = getContentResolver();
Uri base = MediaStore.Audio.Media.EXTERNAL_CONTENT_URI;
Uri newUri = contentResolver.insert(base, values);
// Notify the media application on the device
sendBroadcast(new Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE, newUri));
```

## فرمت های مورد پشتیبانی

اندروید از فرمت های مختلف پشتیبانی می کند. جهت مشاهده ای فرمت های مورد پشتیبانی اندروید می توانید به آدرس <http://developer.android.com/guide/appendix/media-formats.html> مراجعه نمایید.

## آموزش کاربردی: پخش صدا به وسیله ای کلاس SoundPool

در بخش حاضر یک اپلیکیشن خواهیم نوش特 که به مجرد برخورد انگشت کاربر با سطح نمایشگر، یک آهنگ یا صدا را بخش خواهد کرد. ابتدا یک پروژه و activity جدید اندروید به ترتیب به نام های "PlaySound" و "de.vogella.android.soundpool" ایجاد نمایید.

محتوای فایل main.xml که ظاهر اپلیکیشن را تعریف می کند، به صورت زیر ویرایش نمایید.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <TextView
        android:id="@+id/textView1"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="Click on the screen to start playing" >
    </TextView>
</LinearLayout>
```

فایل صوتی مورد نظر را دانلود نمایید. در صورت تمایل می توانید فایل صوتی آزمایشی خود را از آدرس [http://hamsterrepublic.com/ohrrpgce/Free\\_Sound\\_Effects.html](http://hamsterrepublic.com/ohrrpgce/Free_Sound_Effects.html) بارگیری کنید. این سایت تعداد زیادی فایل صوتی کم حجم به صورت رایگان در اختیار شما قرار می دهد. فایل صوتی دریافت شده را داخل پوشه `res/raw` تحت نام "sound1.ogg" ذخیره نمایید. با انتخاب این پسوند برای نام فایل شما به سیستم اعلان می کنید که یک فایل با فرمت ogg از اینترنت بارگیری کرده اید.

کلاس activity خود را به صورت زیر پیاده سازی نمایید.

```
package de.vogella.android.soundpool;
import android.app.Activity;
```

```

import android.media.AudioManager;
import android.media.SoundPool;
import android.media.SoundPool.OnLoadCompleteListener;
import android.os.Bundle;
import android.util.Log;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnTouchListener;
public class PlaySound extends Activity implements OnTouchListener {
    private SoundPool soundPool;
    private int soundID;
    boolean loaded = false;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        View view = findViewById(R.id.textView1);
        view.setOnTouchListener(this);
        // Set the hardware buttons to control the music
        this.setVolumeControlStream(AudioManager.STREAM_MUSIC);
        // Load the sound
        soundPool = new SoundPool(10, AudioManager.STREAM_MUSIC, 0);
        soundPool.setOnLoadCompleteListener(new OnLoadCompleteListener() {
            @Override
            public void onLoadComplete(SoundPool soundPool, int sampleId,
                int status) {
                loaded = true;
            }
        });
        soundID = soundPool.load(this, R.raw.sound1, 1);
    }
    @Override
    public boolean onTouch(View v, MotionEvent event) {
        if (event.getAction() == MotionEvent.ACTION_DOWN) {
            // Getting the user sound settings
            AudioManager audioManager = (AudioManager) getSystemService(AUDIO_SERVICE);
            float actualVolume = (float) audioManager
                .getStreamVolume(AudioManager.STREAM_MUSIC);
            float maxVolume = (float) audioManager
                .getStreamMaxVolume(AudioManager.STREAM_MUSIC);
            float volume = actualVolume / maxVolume;
            // Is the sound loaded already?
            if (loaded) {
                soundPool.play(soundID, volume, volume, 1, 0, 1f);
                Log.e("Test", "Played sound");
            }
        }
        return false;
    }
}

```

حال زمانی که کاربر سطح نمایشگر، activity اپلیکیشن، را لمس می کند، فایل صوتی پخش خواهد شد. لازم به ذکر است که صدا بر اساس تنظیمات صوتی جاری پخش خواهد شد.

## آموزش کاربردی: ضبط صدا (media) با استفاده از کلاس MediaRecorder

در بخش آموزشی حاضر یک اپلیکیشن طراحی می کنیم که به مجرد برخورد انگشت کاربر با سطح نمایشگر، یک فایل صوتی را پخش خواهد می کند. یک پروژه و activity "RecordSound" به نام های "de.vogella.android.media.soundrecording" و "de.vogella.android.media.soundrecording.RecordSound" ایجاد نمایید.

مجوز درج اطلاعات در حافظه خارجی (SD) و ضبط اطلاعات صوتی را در فایل تنظیمات AndroidManifest.xml اعلان نمایید.

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.RECORD_AUDIO" />
```

محتوای فایل main.xml را به صورت زیر اصلاح نمایید.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal" >
    <Button
        android:id="@+id/start"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Start Recording"
        android:onClick="startRecording" />
    <Button
        android:id="@+id/stop"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Stop Recording"
        android:enabled="false"
        android:onClick="stopRecording"
    />
</LinearLayout>
```

کد کلاس activity اپلیکیشن را نیز به صورت زیر ویرایش نمایید.

```
package de.vogella.android.media.soundrecording;
import java.io.File;
import java.io.IOException;
import android.app.Activity;
import android.content.ContentResolver;
import android.content.ContentValues;
import android.content.Intent;
```

```

import android.media.MediaRecorder;
import android.net.Uri;
import android.os.Bundle;
import android.os.Environment;
import android.provider.MediaStore;
import android.util.Log;
import android.view.View;
import android.widget.Toast;
public class SoundRecordingActivity extends Activity {
    MediaRecorder recorder;
    File audiofile = null;
    private static final String TAG = "SoundRecordingActivity";
    private View startButton;
    private View stopButton;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        startButton = findViewById(R.id.start);
        stopButton = findViewById(R.id.stop);
    }
    public void startRecording(View view) throws IOException {
        startButton.setEnabled(false);
        stopButton.setEnabled(true);
        File sampleDir = Environment.getExternalStorageDirectory();
        try {
            audiofile = File.createTempFile("sound", ".3gp", sampleDir);
        } catch (IOException e) {
            Log.e(TAG, "sdcard access error");
            return;
        }
        recorder = new MediaRecorder();
        recorder.setAudioSource(MediaRecorder.AudioSource.MIC);
        recorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);
        recorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
        recorder.setOutputFile(audiofile.getAbsolutePath());
        recorder.prepare();
        recorder.start();
    }
    public void stopRecording(View view) {
        startButton.setEnabled(true);
        stopButton.setEnabled(false);
        recorder.stop();
        recorder.release();
        addRecordingToMediaLibrary();
    }
    protected void addRecordingToMediaLibrary() {
        ContentValues values = new ContentValues(4);
        long current = System.currentTimeMillis();
        values.put(MediaStore.Audio.Media.TITLE, "audio" + audiofile.getName());
        values.put(MediaStore.Audio.Media.DATE_ADDED, (int) (current / 1000));
        values.put(MediaStore.Audio.Media.MIME_TYPE, "audio/3gpp");
    }
}

```

```
values.put(MediaStore.Audio.Media.DATA, audiofile.getAbsolutePath());
ContentResolver contentResolver = getContentResolver();
Uri base = MediaStore.Audio.Media.EXTERNAL_CONTENT_URI;
Uri newUri = contentResolver.insert(base, values);
sendBroadcast(new Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE, newUri));
Toast.makeText(this, "Added File " + newUri, Toast.LENGTH_LONG).show();
}
```

حال زمانی که کاربر دکمه **Start Recording** را فشار می دهد، دستگاه شروع به ضبط صدا می نماید. با فشردن دکمه **stop recording**، می بینید که فرایند ضبط صدا متوقف شده و اطلاعات ذخیره شده متعاقباً به **media library** (مجموعه فایل های صوتی) شما اضافه می گردد.

بخش دوم

# نکات آموزشی در خصوص دریافت و کامپایل کدهای برنامه اندرویدی (source code)

آموزش حاضر به شرح نحوه‌ی دریافت و ترجمه‌ی کدهای اپلیکیشن‌های اندروید و نیز دانلود source code مجموعه‌ی ابزار ساخت و توسعه‌ی اپلیکیشن‌های اندرویدی (ADT) که افزونه‌های plug in های محیط کاری Eclipse هستند، می‌پردازد. این مبحث بر اساس Ubuntu می‌باشد.

## دریافت کد برنامه (Android source code)

جهت دسترسی به کد برنامه‌ی های اندروید شما به دو ابزار Git و repo دارید.

نصب ابزار مورد نیاز

ابتدا لازم است ابزار خط فرمان (command line) Git را نصب نمایید. برای این منظور می‌بایست به آموزش مقدمه‌ای بر Git و نصب آن مراجعه نمایید.

علاوه بر ابزار نام برد، لازم است ابزار دیگری به نام repo را نصب نمایید. برای این منظور کافی است دستورات زیر را اجرا نمایید.

```
# assumes that you have a local directory called bin
# in your home folder
# download the tool from Google
curl http://commondatastorage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
# make it executable
chmod a+x ~/bin/repo
```

## کپی کردن کد اصلی اپلیکیشن (source code cloning)

پس از نصب ابزار repo، شما می توانید کد اپلیکیشن اندرویدی را با فرآخوانی دستور زیر دریافت نمایید.

```
// To get the current master
repo init -u git://android.git.kernel.org/platform/manifest.git ; repo sync ;
```

دستور "git branch -a" به شما تمامی ویرایش های اندروید از جمله froyo و gingerbread را نمایش می دهد. می توانید با درج و اجرای دستور "git checkout branch\_name" تمامی ورژن های اندروید را مشاهده و بررسی نمایید.  
جهت ترجمه و کامپایل کد دستور زیر را فرآخوانی کنید.

make

## ابزار ساخت و توسعه اپلیکیشن های اندرویدی (ADT)

### ابزار ADT

پروژه ای ADT ابزار و تجهیزات نرم افزاری ویژه ای دو محیط برنامه نویسی Eclipse و IntelliJ فراهم می کند که به واسطه ای آن توسعه دهنده می تواند پروژه و برنامه های اندرویدی دلخواه خود را بنویسد.

**نکته:** می توانید برای مشاهده ای اطلاعات و مستندات آموزشی مربوطه به آدرس <http://tools.android.com> مراجعه نمایید.

## کمک به توسعه ی پروژه ADT

فرایند کمک به توسعه ی ADT بر اساس Gerrit می باشد، اما از ابزار دیگری به نام repo استفاده می کند. پروسه ی کمک به توسعه ی پروژه ی مذبور در آدرس <http://tools.android.com/contributing> شرح داده است.

## نحوه ی ساخت ابزار

برای آموزش در خصوص نحوه ی ساخت تجهیزات و ابزار توسعه ADT ویژه ی محیط برنامه نویسی Eclipse می توانید به آدرس زیر مراجعه نمایید:

<http://tools.android.com/build/eclipse>

## بخش سوم

# آموزش ساختار داخلی و معماری سیستم اندروید (Android Architecture)

مقاله ی پیشرو ساختار درونی اندروید را مورد بررسی قرار داده و توضیحاتی در خصوص معماری آن در اختیار شما قرار می دهد.

## برنامه نویسی و توسعه اپلیکیشن برای Android

### سیستم عامل اندروید

اندروید یک سیستم عامل مبتنی بر نسخه ی اصلاح و بروز رسانی شده ی Linux 2.6 و توابع برنامه نویسی و اینترفیس Java می باشد. جهت اجرای بهینه ی اندروید بر روی دستگاه موبایل، تیم توسعه دهندگان اندروید تعدادی کتابخانه و درایور ویرایش شده به آن اضافه کرده اند.

در واقع علاوه بر خود دستگاه مجازی جاوا (Java virtual machine) یا همان DVM، ابزار مختلفی دیگری نظیر کامپایلر، خطایاب کد (debugger) و نرم افزار شبیه ساز (Emulator) در اختیار توسعه دهنده قرار داده شده است.

Android در اصل ساخت تیم نرم افزاری به نام Open Handset Alliance بوده و هم اکنون توسط شرکت گوگل رهبری می شود.

اندروید از یک دستگاه مجازی ویژه به نام Dalvik استفاده می کند. این دستگاه از bytecode های (فرمت کامپایل شده برنامه های جاوا) به خصوصی استفاده می کند، از این جهت شما نمی توانید انتظار اجرای bytecode متعارف جاوا را بر روی محیط اندروید داشته باشید (مجموعه ای از دستورات است که برای اجرای موتور توسعه مفسر، طراحی شده است. بر خلاف کد منبع که برای انسان قابل فهم است، کد قابل حمل نیز خوانده می شود از یک سری کدهای عددی فشرده، آدرس های عددی و ثابت ها تشکیل شده است).

سیستم اندروید ابزاری به نام dx در اختیار توسعه دهنده قرار می دهد که فایل های تشکیل شده از کلاس های جاوا (java class file) را به فایل هایی با پسوند dex (Dalvik Executable) یا فایل های اجرایی dalvik تبدیل می کند. برنامه aapt (ابزار بسته بندی فایل های asset و محتوا در اندروید) اپلیکیشن های اندروید و فایل های آن ها را داخل فایلی میزبانی به نام Android Package با پسوند apk می گنجاند. جهت تسهیل پروسه ای توسعه های پروژه و نرم افزارهای اندرویدی، شرکت گوگل ابزاری تخصصی به نام ADT (مجموعه ابزار ساخت و توسعه اپلیکیشن های اندرویدی) ویژه ای محیط کاری Eclipse ارائه می دهد. ابزار نام برده فرایند تبدیل از کلاس های جاوا به فایل های dex را به صورت خودکار انجام داده و خود فایل apk را در زمان آماده سازی و نصب پروژه (deployment) ایجاد می نماید.

در ادامه مبحث حاضر به شرح سیستم runtime و بستر اجرای اندروید خواهیم پرداخت.

## سیستم داخلی اندروید (internals)

در زمان اجرا و راه اندازی سیستم اندروید، هسته یا kernel لینوکس ابتدا فرایند "init" را فراخوانی می کند. init دو فایل "/init.rc" و "init.device.rc" را به ترتیب می خواند. فایل دوم بر اساس نوع دستگاه متفاوت بوده و به بیانی دقیق تر با توجه به هر دستگاه (device specific) ویژه و اختصاصی می باشد. فایل نام برده در محیط دستگاه مجازی تحت عنوان "init.goldfish.rc" شناخته می شود.

فایل init.rc، به وسیله‌ی ابزاری به نام "/system/bin/app\_process" (یک کامپوننت نرم افزاری و سرویسی از سیستم عامل اندروید که پدر تمامی فرایندهای اپلیکیشن‌های اندرویدی می‌باشد) را اجرا می‌کند. فرایند مجبور کلاس‌های اصلی جاوا را در حافظه بارگذاری نموده و پروسه‌ی پردازش اولیه‌ی آن‌ها را آغاز می‌کند. این کلاس‌ها برای اپلیکیشن‌اندرویدی مورد نظر بارها قابل استفاده بوده (reusable) و همین امر نیز اجرای آن را به مرتب تسریع می‌بخشد. پس از اتمام عملیات اولیه، فرایند ذکر شده (zygote) به socket گوش فرا داده و منتظر دریافت درخواست‌ها می‌شود.

در اندروید درایور ویژه‌ای به نام Binder وجود دارد که زمینه‌ی ارتباطات بین فرایندهای بهینه را فراهم می‌آورد به گونه‌ای که ارجاعات و اشاره‌گرهایی به آبجکت‌ها بین فرایندها رد و بدل می‌شوند و خود آبجکت‌ها در Shared Memory نگهداری می‌شوند. در نتیجه‌ی این امر، از آنجایی که اکنون داده‌ها و اطلاعات کمتری می‌باشد بین فرایندها رد و بدل شود، ارتباط بهینه سازی می‌گردد.

برخلاف سایر سیستم‌های مبتنی بر لینوکس، اندروید فضای جایگزین و اضافی (swap space) در اختیار ندارد، به همین جهت میزان حافظه‌ی قابل استفاده برای سیستم، محدود به همان حافظه‌ی مستقر بر روی دستگاه می‌باشد.

اندروید از کتابخانه‌ی ویژه‌ای C به نام "Bionic" به جای Glibc بهره می‌گیرد. این کتابخانه با Glibc سازگار نبوده و به حافظه کمتری احتیاج دارد. در واقع Bionic در بردارنده‌ی یک نوع خاص از پیاده سازی thread می‌باشد که میزان حافظه‌ی مورد استفاده‌ی هر thread را بهینه ساخته و زمان اجرای thread‌های جدید را کاهش می‌دهد.

از ویرایش 2.3 به بعد، سیستم اندروید از Ext4، سیستم فایل متعارف Linux، استفاده می‌کند. پیش از ویرایش 2.3، سیستم فایل مورد استفاده‌ی اندروید YAFFS بود. برخی از ارائه دهندگان اپلیکیشن و خدمات نرم افزاری (vendor) سیستم فایل اختصاصی و دلخواه خود را جایگزین سیستم فایل متعارف اندروید می‌کنند.

تمامی اپلیکیشن ها user application data اختصاصی خود را دارند و به صورت پیش فرض تنها از خود اپلیکیشن می توان به این فایل ها دسترسی داشت. به منظور به اشتراک گذاری داده ها بین اپلیکیشن ها، شما می توانید اطلاعات را در فایل های ذخیره کنید که برای اپلیکیشن های دیگر قابل دسترسی می باشد. همچنین به عنوان روشی جایگزین، اپلیکیشن ها می توانند content provider اختصاصی خود را تعریف کرده و جهت تامین داده های مورد نیاز خود از این کامپوننت نرم افزاری استفاده کنند. اپلیکیشن هایی که با کلید یا گواهی نامه (certificate) یکسان امضای دیجیتالی شده باشند، می توانند با اعلان المان "android:sharedUserId" در فایل تنظیمات "AndroidManifest.xml"، یک شناسه ی مشترک داشته باشند. اپلیکیشن هایی که شناسه ی مشترک دارند، همگی در بستر فرایند یکسان اجرا شده و شناسه ی کاربری واحدی دارند. از اینرو می توانند به اطلاعات و فایل های هم دسترسی داشته باشند.

فایل apk. اپلیکیشنی که بر روی دستگاه نصب شده در پوشه ی "data/app" جایگذاری می شود. شما می توانید با استفاده از file manager یا دستور adb pull / push، اپلیکیشن را بارگیری کرده یا آن را بر روی دستگاه نصب کنید. فایل نصبی (.apk). اپلیکیشن های پولی یا اپلیکیشن هایی از فروشگاه های مجازی که قابلیت "Copy Protection" آن ها فعال شده، در پوشه ی "data/app-private" قرار دارند و به طور پیش فرض برای کاربر قابل دسترسی نمی باشند.

به منظور محافظت هر چه بیشتر اپلیکیشن در مقابل کپی برداری غیرمجاز، می توانید از کتابخانه "Licence Verification Library" استفاده نمایید. این کتابخانه بررسی می کند آیا اپلیکیشن پولی مورد نظر از Android Market به طور قانونی خریداری شده یا خیر.

در حال حاضر حداقل حجم اپلیکیشن های اندرویدی بین 16 تا 24 مگابایت متغیر می باشد. اینکه حجم واقعی اپلیکیشن چقدر است، منحصرا در زمان کامپایل اپلیکیشن مشخص می شود.

## بخش چهارم

# آموزش C2DM (سرویس اطلاع رسانی (device messaging

آموزش حاضر نحوه‌ی ارسال اطلاعات (push) از یک server (سرویس دهنده) به دستگاه مبتنی بر Google را شرح می‌دهد. پروژه‌ها و مثال‌های این مبحث در محیط کاری Eclipse 3.7، با استفاده از زبان Java نوشته شده و مبتنی بر Android 2.3 می‌باشد.

## Cloud to device messaging

### مقایسه‌ی push و poll

امروزه اغلب اپلیکیشن‌های تحت موبایل اطلاعاتی را از اینترنت دریافت می‌کنند. یک روش برای بروز رسانی اپلیکیشن و دریافت داده‌های جدید این است که اپلیکیشن در فواصل زمانی معین از سرویس دهنده برای داده‌های جدید پرس و جو کند که در اصطلاح به آن polling می‌گویند. چنانچه داده‌های جدیدی برای دانلود موجود نباشد، این روش صرفاً پنهانی باند اضافی و باتری دستگاه را مصرف کرده است.

روش دیگری که می‌توان مورد بررسی قرار داد این است که هر زمان داده‌های جدیدی موجود بود، سرویس دهنده با اپلیکیشن تحت موبایل ارتباط برقرار کرده و آن را از وجود داده‌های جدید آگاه کند که در اصطلاح به آن Pushing می‌گویند. در شرایطی که قرار نیست به طور مداوم داده‌های جدید اضافه شود، توصیه می‌شود که از push استفاده نمایید.

## سرویس ارسال اطلاعات از Server به اپلیکیشن‌های اندرویدی / سرویس Cloud

شما می‌توانید به واسطه‌ی Google play service (سرویس ارائه دهنده خدمات اندروید) اطلاعیه‌هایی را به اپلیکیشن خود ارسال کنید. برای این منظور، لازم است Google Cloud Messaging for Android را از طریق Google API Console فعال نمایید.

در سرویس C2DM سه طرف شرکت دارند: 1. سرویس دهنده ای اپلیکیشن (app server) که پیغام ها و اطلاعاتی را به دستگاه اندروید ارسال (push) می کند. 2. سرورهای Google C2DM 3. اپلیکیشن تحت موبایل اندرویدی. برنامه ای مستقر در سرویس دهنده ای اپلیکیشن می تواند به هر زبانی نوشته شده باشد (Java, PHP, Python و ...).

زمانی که سرویس دهنده ای اپلیکیشن (app server) لازم بداند که پیغام هایی را بایستی به اپلیکیشن تحت موبایل اندرویدی ارسال (push) کند، این پیغام را از طریق متد HTTP POST به سرویس دهنده ای C2DM گوگل ارسال می نماید.

سرورهای C2DM پیغام را به دستگاه مربوطه آدرس دهی و ارسال می کنند (route). چنانچه دستگاه مورد نظر آنلاین نبود، در آن صورت پیغام زمانی تحویل داده می شود که دستگاه به اینترنت وصل و در دسترس قرار گیرد. پس از اینکه پیغام تحویل داده شد (دستگاه آن را دریافت کرد)، یک Broadcast intent ایجاد می شود. اپلیکیشن تحت موبایل از قبل ویژه ای این Intent Receiver، یک Broadcast تعريف کرده است. بنابراین اپلیکیشن مربوطه اجرا شده و پیغام را از طریق Intent Receiver مذبور پردازش می کند.

پیغام های C2DM معمولاً از مرز 1024 بایت تجاوز نمی کنند و منحصراً برای این تعییه شده اند که دستگاه مربوطه را از وجود داده های جدید آگاه کنند. به عبارت دیگر این پیغام با ارسالشان را از وجود اطلاعات جدید آگاه ساخته و هیچ گونه داده ای را به خودی خود به دستگاه منتقل نمی کنند. workflow و روند کار به این صورت هست که سرویس دهنده های C2DM گوگل، اپلیکیشن اندرویدی را از موجود بودن داده های جدید جهت بروز رسانی برنامه مطلع می سازند. پس از آن، اپلیکیشن مستقر بر روی دستگاه موبایل، داده ها را از سرویس دهنده ای دیگری واکشی و دریافت می کند.

دستگاه های اندرویدی همواره ارتباط خود را با سرویس دهنده ای Google Play برقرار نگه می دارند. C2DM از اتصال موجود به سرویس دهنده های گوگل استفاده می کند. این اتصال بهینه سازی شده تا مصرف پهنه ای باند و باتری تا حد امکان کاهش یابد.

در حال حاضر C2DM هنوز در مرحله تست بتا به سر می برد و برای استفاده از آن، شما می بایست درخواست و ثبت نام نمایید. C2DM به هر ارسال کننده اجازه می دهد تا سقف 200,000 پیغام در روز به صورت رایگان ارسال کند.

## ابزار لازم برای تست کاربردی C2DM

سرویس C2MD از ویرایش 2.2 اندروید قابل بهره برداری بوده و جهت استفاده از آن لازم است اپلیکیشن Android Play بر روی دستگاه مورد نظر نصب شده باشد.

برای استفاده از C2DM در شبیه ساز محیط اندروید، بایستی از یک دستگاه که ورژن 8 کتابخانه ای اندروید (API 8) یا بالاتر بر روی آن نصب است استفاده نموده و نیز از طریق بخش Settings، یک حساب کاربری Google در شبیه ساز ایجاد نمایید.

## مجوزهای لازم

جهت استفاده از C2DM در اپلیکیشن خود، بایستی مجوزهای زیر را اعلان نمایید:

- com.google.android.c2dm.permission.RECEIVE
- android.permission.INTERNET

علاوه بر دو مجوز مزبور، اپلیکیشن شما بایستی مجوز "applicationPackage" که با "signature" را با "android:protectionLevel=". را با "android:protectionLevel=C2D\_MESSAGE" مقداردهی شده تنظیم نمایید تا بدین وسیله سایر اپلیکیشن ها نتوانند به پیغام گوش داده و آن را دریافت کنند. در واقع اپلیکیشن های دیگر برای اینکه بتوانند به این پیغام گوش داده و آن را دریافت کنند، می بایست با گواهی نامه و امضای دیجیتالی (certificate) که اپلیکیشن اصلی دریافت کننده می باشد، امضا شده باشند. به عبارت دیگر مجوز "applicationPackage" می شود تنها اپلیکیشن هایی که با امضای دیجیتالی یکسان امضا شده اند، بتوانند پیغام را دریافت کنند.

## اعلان intent receiver جهت دریافت intent های (Intent Receiver)

### مربوطه

اپلیکیشن شما می بایست یک intent receiver جهت دریافت دو intent اعلان نماید:

- com.google.android.c2dm.intent.REGISTRATION

- com.google.android.c2dm.intent.RECEIVE

ای که ویژه‌ی "com.google.android.c2dm.intent.RECEIVE" اعلان شده، زمانی صدا زده می شود که پیغام جدیدی دریافت شده باشد، در حالی که receiver اعلان شده برای "com.google.android.c2dm.intent.REGISTRATION" زمانی فراخوانی می شود که کد ثبت (registration code) اپلیکیشن مربوطه دریافت شده باشد.

## مراحل پیاده سازی

### ثبت و معرفی app server (سرвис دهنده‌ی اپلیکیشن)

سرвис دهنده‌ی اپلیکیشن می بایست خود را برای سرورهای C2DM معرفی و احراز هویت (authenticate) نماید. از طریق ایمیل و گذرواژه، یک شناسه‌ی احراز هویت یا امنیتی (authentication token) که با متد درخواست اطلاعات HTTP POST به سرورهای C2DM تعیین شده و در اختیار اپلیکیشن قرار می گیرد. حال شناسه‌ی امنیتی یا همان token در اپلیکیشن سمت سرویس دهنده (app server) ذخیره شده و زمانی که اپلیکیشن می خواهد پیغامی را ارسال کند، خود را به واسطه‌ی این شناسه به سرورهای C2DM می شناساند.

برای مثال، جهت دریافت شناسه‌ی امنیتی که با ارائه‌ی آدرس و گذرواژه‌ی ایمیل معتبر و ثبت شده، در اختیار اپلیکیشن قرار می گیرد، می توانید به صورت زیر اقدام نمایید (کد و کلاس زیر را پیاده سازی نمایید):

```
package de.vogella.java.c2dm.server.util;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.URL;
```

```

public class AuthenticationUtil {
    private AuthenticationUtil() {
        // Util class cannot get instantiated
    }
    public static String getToken(String email, String password)
        throws IOException {
        // create the post data
        // Requires a field with the email and the password
        StringBuilder builder = new StringBuilder();
        builder.append("Email=").append(email);
        builder.append("&Passwd=").append(password);
        builder.append("&accountType=GOOGLE");
        builder.append("&source=MyLittleExample");
        builder.append("&service=ac2dm");
        // Setup the Http Post
        byte[] data = builder.toString().getBytes();
        URL url = new URL("https://www.google.com/accounts/ClientLogin");
        HttpURLConnection con = (HttpURLConnection) url.openConnection();
        con.setUseCaches(false);
        con.setDoOutput(true);
        con.setRequestMethod("POST");
        con.setRequestProperty("Content-Type",
            "application/x-www-form-urlencoded");
        con.setRequestProperty("Content-Length", Integer.toString(data.length));
        // Issue the HTTP POST request
        OutputStream output = con.getOutputStream();
        output.write(data);
        output.close();
        // read the response
        BufferedReader reader = new BufferedReader(new InputStreamReader(
            con.getInputStream()));
        String line = null;
        String auth_key = null;
        while ((line = reader.readLine()) != null) {
            if (line.startsWith("Auth=")) {
                auth_key = line.substring(5);
            }
        }
        // Finally get the authentication token
        // To something useful with it
        return auth_key;
    }
}

```

شناسه‌ی امنیتی (token) در فواصل زمانی معین بروز رسانی می‌شود.

مثال بالا با زبان جاوا نوشته شده است. این امکان نیز وجود دارد که شناسه‌ی مورد نیاز را با استفاده از سایر ابزار http یا زبان‌های برنامه نویسی تهیه کرد. به طور مثال، شما می‌توانید با بهره‌گیری از ابزار خط فرمان (command line tool) و ابزار curl، سرور را شبیه سازی نمایید.

## دریافت شناسه‌ی ثبت (registration ID) اپلیکیشن تحت موبایل

جهت معرفی اپلیکیشن های اندرویدی و تحت موبایل خود به سرورهای C2DM (به منظور دریافت پیغام و خدمات بروز رسانی از سرور توسعه دهنده اپلیکیشن)، لازم است فراخوانی (trigger) شده که شناسه‌ی ثبت (registration id) را به سرورهای C2DM ارسال می‌نماید.

آبجکت intent حامل اطلاعات اضافی به نام "sender" به عنوان کلید (پارامتر ورودی اول متد putExtra) و آدرس ایمیل که با آن ثبت انجام شده به عنوان پارامتر ورودی دوم پذیرفته می‌شود. بعلاوه، آبجکت intent مجبور بایستی دربردارنده‌ی PendingIntent با اطلاعات اضافی در قالب extra که به عنوان پارامتر اول به متد putExtra ارسال شده، باشد. به سیستم اندروید اطلاعاتی پیرامون اپلیکیشن جاری ارائه می‌دهد. مقدار sender آدرس ایمیلی است که تحت آن سرویس پیغام رسانی C2MD خود را ثبت یا اعلان نمودید. مقدار رشته‌ای putExtra "youruser@gmail.com" را که آدرس ایمیل شما می‌باشد، جایگزین پارامتر دوم متد putExtra نمایید.

```
public void register(View view) {
    Intent intent = new Intent("com.google.android.c2dm.intent.REGISTER");
    intent.putExtra("app",PendingIntent.getBroadcast(this, 0, new Intent(), 0));
    intent.putExtra("sender", "youruser@gmail.com");
    startService(intent);
}
```

سرویس به صورت ناهمzman خود را به Google معرفی کرده و اینترنت (registration) ارسال می‌کند. اپلیکیشن شما بایستی یک Broadcast receiver ویژه‌ی این intent ثبت کند. برای این منظور لازم است مجوز مورد نیاز را بر اساس پکیج خود اعلان نمایید چرا که سیستم اندروید این امر را از داخل بررسی می‌کند.

```
package de.vogella.android.c2dm.simpleclient;
import android.content.BroadcastReceiver;
import android.content.Context;
```

```

import android.content.Intent;
import android.util.Log;
public class C2DMRegistrationReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();
        Log.w("C2DM", "Registration Receiver called");
        if ("com.google.android.c2dm.intent.REGISTRATION".equals(action)) {
            Log.w("C2DM", "Received registration ID");
            final String registrationId = intent
                .getStringExtra("registration_id");
            String error = intent.getStringExtra("error");
            Log.d("C2DM", "dmControl: registrationId = " + registrationId
                + ", error = " + error);
            // TODO Send this to my application server
        }
    }
}

```

محتوای فایل تنظیمات اپلیکیشن "AndroidManifest.xml" مشابه زیر می باشد. به خاطر داشته باشید که اگر از پکیج دیگری استفاده می کنید، بایستی کد و محتوای فایل را نیز متعاقباً تنظیم نمایید.

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="de.vogella.android.c2dm.simpleclient"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk android:minSdkVersion="8" />
    <permission
        android:name="de.vogella.android.c2dm.simpleclient.permission.C2D_MESSAGE"
        android:protectionLevel="signature" />
    <uses-permission
        android:name="de.vogella.android.c2dm.simpleclient.permission.C2D_MESSAGE" />
    <uses-permission android:name="com.google.android.c2dm.permission.RECEIVE" />
    <uses-permission android:name="android.permission.INTERNET" />
    <application
        android:icon="@drawable/icon"
        android:label="@string/app_name" >
        <activity
            android:label="@string/app_name"
            android:name=".C2DMClientActivity" >
            <intent-filter >
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <receiver
            android:name=".C2DMRegistrationReceiver"
            android:permission="com.google.android.c2dm.permission.SEND" >

```

```

<intent-filter>
    <action android:name="com.google.android.c2dm.intent.REGISTRATION" />
    </action>
    <category android:name="de.vogella.android.c2dm.simpleclient" />
</intent-filter>
</receiver>
</application>
</manifest>

```

اینترنت "com.google.android.c2dm.intent.REGISTRATION" دربردارنده یک شناسه ی ثبت (registration ID) می باشد. هر registration ID نشانگر و نماینده یک دستگاه منحصر بفرد و معین می باشد. مانند اینکه هر گوشی اندروید یک کد ثبت و شناسه ی (registration code) اختصاصی خود را دارد.

ممکن است C2DM شناسه ی ثبت را در فواصل زمانی معین بروز رسانی کند اما تا آن زمان، اپلیکیشن شما می بایست این ID را برای استفاده ی آینده نزد خود ذخیره نگه دارد.

پس از اینکه اپلیکیشن اندرویدی شناسه ی ثبت (registration ID) را دریافت کرد، می بایست این اطلاعات را به سرویس دهنده ی اپلیکیشن (App server) ارسال کند. این سرویس دهنده با استفاده از شناسه ی ثبت، یک پیغام از طریق سرورهای C2DM به دستگاهی که اپلیکیشن اندرویدی بر روی آن نصب است ارسال می کند.

به عنوان مثال، کد زیر deviceId و registrationId را به یک سرویس دهنده ارسال می کند.

```

// Better do this in an asynchronous thread
public void sendRegistrationIdToServer(String deviceId, String registrationId) {
    Log.d("C2DM", "Sending registration ID to my application server");
    HttpClient client = new DefaultHttpClient();
    HttpPost post = new HttpPost("http://your_url/register");
    try {
        List<NameValuePair> nameValuePairs = new ArrayList<NameValuePair>(1);
        // Get the deviceID
        nameValuePairs.add(new BasicNameValuePair("deviceid", deviceId));
        nameValuePairs.add(new BasicNameValuePair("registrationid", registrationId));
        post.setEntity(new UrlEncodedFormEntity(nameValuePairs));
        HttpResponse response = client.execute(post);
        BufferedReader rd =

```

```

        new BufferedReader(new InputStreamReader(response.getEntity().getContent()));
        String line = "";
        while ((line = rd.readLine()) != null) {
            Log.e("Http Response", line);
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}
}

```

سرویس دهنده ID registration ها را به طور دائمی در خود ذخیره می کند.

## ثبت و اعلان یک Receiver ویژه ی پیغام های ارسالی از سرورهای C2DM

درست مشابه ثبت message receiver، بایستی یک registration receiver جهت دریافت پیغام های ارسالی از C2MD معرفی نمایید. این می تواند همان registration receiver یا مجزا باشد. مثال زیر ثبت یک message receiver مجزا را نشان می دهد.

```

package de.vogella.android.c2dm.simpleclient;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.util.Log;
public class C2DMMessagesReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();
        Log.w("C2DM", "Message Receiver called");
        if ("com.google.android.c2dm.intent.RECEIVE".equals(action)) {
            Log.w("C2DM", "Received message");
            final String payload = intent.getStringExtra("payload");
            Log.d("C2DM", "dmControl: payload = " + payload);
            // Send this to my application server
        }
    }
}

```

همچنین لازم است که message receiver را به صورت زیر در فایل تنظیمات اپلیکیشن اعلان نمایید. AndroidManifest.xml

```

<receiver android:name=".C2DMMessagesReceiver"
    android:permission="com.google.android.c2dm.permission.SEND">
    <intent-filter>
        <action android:name="com.google.android.c2dm.intent.RECEIVE"></action>
        <category android:name="de.vogella.android.c2dm.simpleclient" />
    </intent-filter>
</receiver>

```

## ارسال پیغام

اکنون application server و اپلیکیشن تحت موبایل اندرویدی شما آماده‌ی استفاده از C2DM و تبادل اطلاعات با یکدیگر هستند. سرور، شناسه‌ی امنیتی/احراز هویت (authentication) و شناسه‌ی ثبت (registration ID) اپلیکیشن سرویس گیرنده را در اختیار دارد و اپلیکیشن نیز با اعلان Broadcast Receiver آماده‌ی این است که پیغام‌های مربوطه را دریافت کند.

به منظور ارسال پیغام به دستگاه مورد نظر (سرویس گیرنده‌ی نهایی)، یک App server درخواست HTTP POST به سرورهای C2MD گوگل ارسال می‌کند. این درخواست یا متد HTTP GET در بردارنده‌ی registration ID دستگاه مورد نظر و شناسه‌ی احراز هویت (برای اینکه به گوگل اعلان کند که سرور اجازه‌ی ارسال پیغام به اپلیکیشن تحت موبایل را دارد) می‌باشد.

```
package de.vogella.java.c2dm.server.util;
import java.io.IOException;
import java.io.OutputStream;
import java.net.URL;
import java.net.URLEncoder;
import javax.net.ssl.HostnameVerifier;
import javax.net.ssl.HttpsURLConnection;
import javax.net.ssl.SSLSession;
public class MessageUtil {
    private final static String AUTH = "authentication";
    private static final String UPDATE_CLIENT_AUTH = "Update-Client-Auth";
    public static final String PARAM_REGISTRATION_ID = "registration_id";
    public static final String PARAM_DELAY_WHILE_IDLE = "delay_while_idle";
    public static final String PARAM_COLLAPSE_KEY = "collapse_key";
    private static final String UTF8 = "UTF-8";
    public static int sendMessage(String auth_token, String registrationId,
        String message) throws IOException {
        StringBuilder postDataBuilder = new StringBuilder();
        postDataBuilder.append(PARAM_REGISTRATION_ID).append("=")
            .append(registrationId);
        postDataBuilder.append("&").append(PARAM_COLLAPSE_KEY).append("=")
            .append("0");
        postDataBuilder.append("&").append("data.payload").append("=")
            .append(URLEncoder.encode(message, UTF8));
        byte[] postData = postDataBuilder.toString().getBytes(UTF8);
        // Hit the dm URL.
        URL url = new URL("https://android.clients.google.com/c2dm/send");
        HttpsURLConnection
            .setDefaultHostnameVerifier(new CustomizedHostnameVerifier());
```

```

HttpsURLConnection conn = (HttpsURLConnection) url.openConnection();
conn.setDoOutput(true);
conn.setUseCaches(false);
conn.setRequestMethod("POST");
conn.setRequestProperty("Content-Type",
        "application/x-www-form-urlencoded;charset=UTF-8");
conn.setRequestProperty("Content-Length",
        Integer.toString(postData.length));
conn.setRequestProperty("Authorization", "GoogleLogin auth="
        + auth_token);
OutputStream out = conn.getOutputStream();
out.write(postData);
out.close();
int responseCode = conn.getResponseCode();
return responseCode;
}
private static class CustomizedHostnameVerifier implements HostnameVerifier {
    public boolean verify(String hostname, SSLSession session) {
        return true;
    }
}
}

```

زمانی که سرویس دهنده (app server) پیغام را به سرور C2DM ارسال می کند، این سرور پیغام را در صفحه و حالت انتظار نگه می دارد تا دستگاه آنلاین شود. پیغام در قالب یک broadcast به دستگاه مربوطه ارسال می شود. اپلیکیشن شما بایستی از قبل برای این broadcast event ثبت نام کرده (به آن گوش فرا داده) تا آن را به هنگام ارسال، دریافت نماید.

پیغام دریافتی به broadcast receiver ای که به "com.google.android.c2dm.intent.RECEIVE" گوش فرا داده و ویژه آن ثبت شده، فرستاده می شود. داده های مورد نظر به راحتی با فراخوانی متد () getExtras بر روی نمونه می ساخته شده از کلاس Intent قابل بازیابی می باشد. کلیدهای موجود عبارتند از "payload" ، "collapse\_key" ، "from" . داده های اصلی در پارامتر "payload" قرار دارند. receiver می تواند این داده ها را استخراج کرده و به آن واکنش مناسب را نشان دهد.

## ثبت نام و درخواست برای استفاده از سرویس C2DM

در حال حاضر C2DM در مرحله ی آزمایش بتا به سر می برد. برای استفاده از آن نیاز به مجوز دسترسی و ثبت نام دارید. برای ثبت نام و دسترسی به این سرویس می توانید به [code.google.com/android/c2dm/signup.html](http://code.google.com/android/c2dm/signup.html) مراجعه نمایید.

اگر می خواهید مثال زیر را بر روی محیط شبیه ساز اجرا و تست نمایید، در آن صورت لازم است ورژن 8 کتابخانه های اندروید (API 8) یا بالاتر را نصب کرده و در اختیار داشته باشید. همچنین لازم است یک حساب کاربری گوگل (Google user) در محیط شبیه ساز ایجاد نمایید. برای این منظور کافی است مسیر رو به رو را طی کنید: **Settings ▶ Accounts Sync**.

چنانچه قصد دارید که مثال حاضر را بر روی دستگاه واقعی اندروید تست نمایید، در آن صورت لازم است Android Market را بر روی دستگاه مربوطه نصب کنید.

## آموزش کاربردی: طراحی و ساخت اپلیکیشن با قابلیت تعامل با سرورهای C2DM (C2DM enabled)

### ایجاد پروژه و فایل layout

"de.vogella.android.c2dm.simpleclient" activity جدید دیگری به ترتیب به نام های "C2DMClientActivity" و "main.xml" ایجاد نمایید. حال یک فایل main.xml با محتوای زیر ایجاد نمایید.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="register"
        android:text="Register" >
    </Button>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="showRegistrationId"
        android:text="Show" >
```

953

آدرس آموزشگاه : تهران - خیابان شریعتی - بالاتر از خیابان ملک - جنب بانک صادرات - پلاک 651 طبقه دوم - واحد 7

88146323 - 88446780 - 88146330

```
</Button>
</LinearLayout>
```

سپس فایل "activity\_result.xml" را با محتوای زیر ایجاد نمایید. این فایل را در activity های که صفحه‌ی نتایج را نمایش می‌دهند فراخوانی خواهیم کرد.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <TextView
        android:id="@+id/result"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_gravity="center"
        android:text="No info."
        android:textAppearance="?android:attr/textAppearanceLarge" >
    </TextView>
</LinearLayout>
```

## ایجاد activity ها و receiver های مورد نیاز

دو کلاس زیر به نام های C2DMMessagesReceiver و C2DMRegistrationReceiver را ایجاد نمایید. این دو کلاس بعده ها به عنوان receiver یا گوش فرادهنده به (دریافت کننده) معرفی شده و مورد استفاده قرار خواهند گرفت.

```
package de.vogella.android.c2dm.simpleclient;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.List;
import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.HttpClient;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicNameValuePair;
import android.app.Notification;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.SharedPreferences.Editor;
import android.preference.PreferenceManager;
```

```

import android.provider.Settings.Secure;
import android.util.Log;
public class C2DMRegistrationReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();
        Log.w("C2DM", "Registration Receiver called");
        if ("com.google.android.c2dm.intent.REGISTRATION".equals(action)) {
            Log.w("C2DM", "Received registration ID");
            final String registrationId = intent
                .getStringExtra("registration_id");
            String error = intent.getStringExtra("error");
            Log.d("C2DM", "dmControl: registrationId = " + registrationId
                + ", error = " + error);
            String deviceld = Secure.getString(context.getContentResolver(),
                Secure.ANDROID_ID);
            createNotification(context, registrationId);
            sendRegistrationIdToServer(deviceld, registrationId);
            // Also save it in the preference to be able to show it later
            saveRegistrationId(context, registrationId);
        }
    }

    private void saveRegistrationId(Context context, String registrationId) {
        SharedPreferences prefs = PreferenceManager
            .getDefaultSharedPreferences(context);
        Editor edit = prefs.edit();
        edit.putString(C2DMClientActivity.AUTH, registrationId);
        edit.commit();
    }

    public void createNotification(Context context, String registrationId) {
        NotificationManager notificationManager = (NotificationManager) context
            .getSystemService(Context.NOTIFICATION_SERVICE);
        Notification notification = new Notification(R.drawable.icon,
            "Registration successful", System.currentTimeMillis());
        notification.flags |= Notification.FLAG_AUTO_CANCEL;
        Intent intent = new Intent(context, RegistrationResultActivity.class);
        intent.putExtra("registration_id", registrationId);
        PendingIntent pendingIntent = PendingIntent.getActivity(context, 0,
            intent, 0);
        notification.setLatestEventInfo(context, "Registration",
            "Successfully registered", pendingIntent);
        notificationManager.notify(0, notification);
    }

    // incorrect usage as the receiver may be canceled at any time
    // do this in an service and in an own thread
    public void sendRegistrationIdToServer(String deviceld,
        String registrationId) {
        Log.d("C2DM", "Sending registration ID to my application server");
        HttpClient client = new DefaultHttpClient();
        HttpPost post = new HttpPost("http://vogellac2dm.appspot.com/register");
        try {

```

```

List<NameValuePair> nameValuePairs = new ArrayList<NameValuePair>(1);
// Get the deviceID
nameValuePairs.add(new BasicNameValuePair("deviceid", deviceId));
nameValuePairs.add(new BasicNameValuePair("registrationid",
registrationId));
post.setEntity(new UrlEncodedFormEntity(nameValuePairs));
HttpResponse response = client.execute(post);
BufferedReader rd = new BufferedReader(new InputStreamReader(
response.getEntity().getContent()));
String line = "";
while ((line = rd.readLine()) != null) {
Log.e("HttpResponse", line);
}
} catch (IOException e) {
e.printStackTrace();
}
}
}
}

package de.vogella.android.c2dm.simpleclient;
import android.app.Notification;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.util.Log;
public class C2DMMessagesReceiver extends BroadcastReceiver {
@Override
public void onReceive(Context context, Intent intent) {
String action = intent.getAction();
Log.w("C2DM", "Message Receiver called");
if ("com.google.android.c2dm.intent.RECEIVE".equals(action)) {
Log.w("C2DM", "Received message");
final String payload = intent.getStringExtra("payload");
Log.d("C2DM", "dmControl: payload = " + payload);
// TODO Send this to my application server to get the real data
// Lets make something visible to show that we received the message
createNotification(context, payload);
}
}
}

public void createNotification(Context context, String payload) {
NotificationManager notificationManager = (NotificationManager) context
.getSystemService(Context.NOTIFICATION_SERVICE);
Notification notification = new Notification(R.drawable.icon,
"Message received", System.currentTimeMillis());
// hide the notification after its selected
notification.flags |= Notification.FLAG_AUTO_CANCEL;
Intent intent = new Intent(context, MessageReceivedActivity.class);
intent.putExtra("payload", payload);
PendingIntent pendingIntent = PendingIntent.getActivity(context, 0,
intent, PendingIntent.FLAG_CANCEL_CURRENT);
}

```

```

        notification.setLatestEventInfo(context, "Message",
            "New message received", pendingIntent);
        notificationManager.notify(0, notification);

    }

}

```

حال دو کلاس activity زیر که نتایج را در صفحه برای کاربر به نمایش می گذارند، ایجاد و پیاده سازی نمایید.

```

package de.vogella.android.c2dm.simpleclient;
import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;
public class RegistrationResultActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        setContentView(R.layout.activity_result);
        Bundle extras = getIntent().getExtras();
        if (extras != null) {
            String registrationId = extras.getString("registration_id");
            if (registrationId != null && registrationId.length() > 0) {
                TextView view = (TextView) findViewById(R.id.result);
                view.setText(registrationId);
            }
        }
        super.onCreate(savedInstanceState);
    }
}

```

```

package de.vogella.android.c2dm.simpleclient;
import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;
public class MessageReceivedActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        setContentView(R.layout.activity_result);
        Bundle extras = getIntent().getExtras();
        if (extras != null) {
            String message = extras.getString("payload");
            if (message != null && message.length() > 0) {
                TextView view = (TextView) findViewById(R.id.result);
                view.setText(message);
            }
        }
        super.onCreate(savedInstanceState);
    }
}

```

فایل تنظیمات اپلیکیشن AndroidManifest.xml را به صورت زیر اعلان نمایید. این فایل activity ها را تعریف کرده و مجوزهای لازم را درخواست می کند.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="de.vogella.android.c2dm.simpleclient"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk android:minSdkVersion="8" />
    <permission
        android:name="de.vogella.android.c2dm.simpleclient.permission.C2D_MESSAGE"
        android:protectionLevel="signature" />
    <uses-permission
        android:name="de.vogella.android.c2dm.simpleclient.permission.C2D_MESSAGE" />
    <uses-permission android:name="com.google.android.c2dm.permission.RECEIVE" />
    <uses-permission android:name="android.permission.INTERNET" />
    <application
        android:icon="@drawable/icon"
        android:label="@string/app_name" >
        <activity
            android:label="@string/app_name"
            android:name=".C2DMClientActivity" >
            <intent-filter >
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <receiver
            android:name=".C2DMRegistrationReceiver"
            android:permission="com.google.android.c2dm.permission.SEND" >
            <intent-filter >
                <action android:name="com.google.android.c2dm.intent.REGISTRATION" />
                <category android:name="de.vogella.android.c2dm.simpleclient" />
            </intent-filter>
        </receiver>
        <receiver
            android:name=".C2DMMessagesReceiver"
            android:permission="com.google.android.c2dm.permission.SEND" >
            <intent-filter >
                <action android:name="com.google.android.c2dm.intent.RECEIVE" />
                <category android:name="de.vogella.android.c2dm.simpleclient" />
            </intent-filter>
        </receiver>
        <activity android:name="RegistrationResultActivity" >
        </activity>
        <activity android:name="MessageReceivedActivity" >
        </activity>
    </application>
```

&lt;/manifest&gt;

## محتوای کلاس "C2DMClientActivity" را به صورت زیر ویرایش نمایید.

```

package de.vogella.android.c2dm.simpleclient;
import android.app.Activity;
import android.app.PendingIntent;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.util.Log;
import android.view.View;
import android.widget.Toast;
public class C2DMClientActivity extends Activity {
    public final static String AUTH = "authentication";
    // Example Activity to trigger a request for a registration ID to the Google
    // server
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
    public void register(View view) {
        Log.w("C2DM", "start registration process");
        Intent intent = new Intent("com.google.android.c2dm.intent.REGISTER");
        intent.putExtra("app",
        PendingIntent.getBroadcast(this, 0, new Intent(), 0));
        // Sender currently not used
        intent.putExtra("sender", "nonsense@gmail.com");
        startService(intent);
    }
    public void showRegistrationId(View view) {
        SharedPreferences prefs = PreferenceManager
        .getDefaultSharedPreferences(this);
        String string = prefs.getString(AUTH, "n/a");
        Toast.makeText(this, string, Toast.LENGTH_LONG).show();
        Log.d("C2DM RegId", string);
    }
}

```

متدهای موجود در activity به واسطه‌ی خاصیت onclick (property) به المان‌های button و registration ID متصل هستند. اولین دکمه در activity، یک درخواست برای ID در لایه‌ی XML متصل است. دومین دکمه کلید ثبت یا registration key ذخیره شده را نمایش می‌دهد. هر دو دکمه دومین دکمه کلید ثبت یا registration key را در registration id Logcat view (logcat view) چاپ می‌کنند.

کپی نمایید چرا که بعده ها می بایست آن را در پیاده سازی Logcat View را از Registration id سرور مجدداً مورد استفاده قرار دهید.

## ثبت اپلیکیشن برای دریافت پیغام (Registration)

اپلیکیشن خود را اجرا کرده، حساب کاربری ثبت شده را نزد خود نگاه داشته و سپس دکمه‌ی مورد نظر را فشار دهید. حال LogCat ID را در registration استفاده کنید.

در صورت مشاهده‌ی پیغام زیر مطمئن شوید که از یک دستگاه مبتنی بر Google استفاده نموده و یک حساب کاربری Google در دستگاه مربوطه ایجاد کرده‌اید:

```
Unable to start service Intent
{act=com.google.android.c2dm.intent.REGISTER ... }: not found
```

## آموزش کاربردی: استفاده از ابزار خط فرمان curl جهت شبیه سازی قابلیت‌های سرور

سیستم عامل‌های مبتنی بر Linux این امکان را به شما می‌دهنده تا سرویس مورد نظر را به راحتی از طریق خط فرمان (command line) تست نمایید. می‌توان با استفاده از curl در خط فرمان، کلید احراز هویت (authentication key) را درخواست نمود. از پاسخ/خروجی، آن بخشی که پس از "Auth=" قرار دارد را بازیابی نمایید.

```
curl https://www.google.com/accounts/ClientLogin -d
  Email=your_user -d "Passwd=your_password" -d accountType=GOOGLE
  -d source=Google-cURL-Example -d service=ac2dm
```

این بخش (قطعه‌ی پس از auth) و کد ثبت (registration code) را می‌توان جهت ارسال یک پیغام به دستگاه مربوطه مورد استفاده قرار داد.

```
curl --header "Authorization: GoogleLogin auth=your_authenticationid"
  "https://android.apis.google.com/c2dm/send" -d registration_id=your_registration
  -d "data.payload=payload" -d collapse_key=0
```

## آموزش کاربردی: ایجاد server app

همان طور که قبلا تشریح شد، app server می باشد از طریق پروتکل HTTPS یک Authentication key (کلیدی جهت احراز هویت) بازیابی کند. پس از آن app server قادر خواهد بود که پیغام هایی را از طریق پروتکل HTTP به دستگاه مربوطه ارسال نماید. app server جهت شناساندن خود به سرور C2MD می باشد کلید احراز هویت (Auth key) و شناسه‌ی ثبت اپلیکیشن (registration ID) را ارائه نماید.

در تمرین حاضر قصد داریم تا سرور را به واسطه‌ی یک برنامه‌ی نوشته شده با جاوا شبیه سازی کنیم. از آنجایی که امکان برقراری ارتباط اپلیکیشن از طریق پروتکل HTTP برای ما وجود ندارد، شناسه‌ی ثبت (registration ID) دستگاه به صورت hardcoded در اپلیکیشن قرار دارد. به خاطر داشته باشید این اپلیکیشن صرفاً یک مثال بوده و جهت تست آسان C2DM طراحی شده است.

جهت ذخیره‌ی شناسه‌ی کاربری و اعتبارنامه (credentials)، کلاس زیر را پیاده سازی نمایید.

```
package de.vogella.java.c2dm.server.secret;
public class SecureStorage {
    public static final String USER = "your_registeredUser";
    public static final String PASSWORD = "your_password";
}
```

یک پروژه‌ی جدید جاوایی به نام "de.vogella.java.c2dm.server" ایجاد نمایید. سپس کلاس زیر را پیاده سازی کنید. این کلاس در واقع یک کلاس کمکی (utility class) است که به منظور بازیابی شناسه‌ی امنیتی/احراز هویت (authentication token) از سرور Google مورد استفاده قرار می‌گیرد.

```
package de.vogella.java.c2dm.server.util;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.URL;
public class AuthenticationUtil {
    private AuthenticationUtil() {
        // Util class cannot get instanziated
    }
    public static String getToken(String email, String password)
        throws IOException {
        // create the post data
```

```

// Requires a field with the email and the password
StringBuilder builder = new StringBuilder();
builder.append("Email=").append(email);
builder.append("&Passwd=").append(password);
builder.append("&accountType=GOOGLE");
builder.append("&source=MyLittleExample");
builder.append("&service=ac2dm");
// Setup the Http Post
byte[] data = builder.toString().getBytes();
URL url = new URL("https://www.google.com/accounts/ClientLogin");
HttpURLConnection con = (HttpURLConnection) url.openConnection();
con.setUseCaches(false);
con.setDoOutput(true);
con.setRequestMethod("POST");
con.setRequestProperty("Content-Type",
    "application/x-www-form-urlencoded");
con.setRequestProperty("Content-Length", Integer.toString(data.length));
// Issue the HTTP POST request
OutputStream output = con.getOutputStream();
output.write(data);
output.close();
// read the response
BufferedReader reader = new BufferedReader(new InputStreamReader(
    con.getInputStream()));
String line = null;
String auth_key = null;
while ((line = reader.readLine()) != null) {
    if (line.startsWith("Auth=")) {
        auth_key = line.substring(5);
    }
}
// Finally get the authentication token
// To something useful with it
return auth_key;
}
}

```

کلاسی به نام GetAuthenticationToken را با بدنه‌ی زیر پیاده سازی نمایید. کلاس نام برده قادر است شناسه‌ی امنیتی (auth token) را واکشی کند.

```

package de.vogella.java.c2dm.server;
import java.io.IOException;
import de.vogella.java.c2dm.server.secret.SecureStorage;
import de.vogella.java.c2dm.server.util.AuthenticationUtil;
public class GetAuthenticationToken {
    public static void main(String[] args) throws IOException {
        String token = AuthenticationUtil.getToken(SecureStorage.USER,
            SecureStorage.PASSWORD);
        System.out.println(token);
    }
}

```

کلاس `GetAuthenticationToken` را اجرا نموده و شناسه‌ی امنیتی (`authentication token`) را از خط فرمان کپی نمایید. حال کلاس زیر را تعریف کرده، شناسه‌ی امنیتی (`authentication`) و شناسه‌ی ثبت اپلیکیشن (`registration id`) را نزد خود نگاه دارید.

```
package de.vogella.java.c2dm.server;
public class ServerConfiguration {
    public static final String AUTHENTICATION_TOKEN = "your_token";
    public static final String REGISTRATION_ID = "registration_id_of_your_device";
}
```

سپس یک کلاس کمکی (`utility class`) جهت ارسال پیغام به دستگاه مربوطه با پیاده‌سازی زیر ایجاد نمایید.

```
package de.vogella.java.c2dm.server.util;
import java.io.IOException;
import java.io.OutputStream;
import java.net.URL;
import java.net.URLEncoder;
import javax.net.ssl.HostnameVerifier;
import javax.net.ssl.HttpsURLConnection;
import javax.net.ssl.SSLSession;
public class MessageUtil {
    private final static String AUTH = "authentication";
    private static final String UPDATE_CLIENT_AUTH = "Update-Client-Auth";
    public static final String PARAM_REGISTRATION_ID = "registration_id";
    public static final String PARAM_DELAY_WHILE_IDLE = "delay_while_idle";
    public static final String PARAM_COLLAPSE_KEY = "collapse_key";
    private static final String UTF8 = "UTF-8";
    public static int sendMessage(String auth_token, String registrationId,
        String message) throws IOException {
        StringBuilder postDataBuilder = new StringBuilder();
        postDataBuilder.append(PARAM_REGISTRATION_ID).append("=")
            .append(registrationId);
        postDataBuilder.append("&").append(PARAM_COLLAPSE_KEY).append("=")
            .append("0");
        postDataBuilder.append("&").append("data.payload").append("=")
            .append(URLEncoder.encode(message, UTF8));
        byte[] postData = postDataBuilder.toString().getBytes(UTF8);
        // Hit the dm URL.
        URL url = new URL("https://android.clients.google.com/c2dm/send");
        HttpsURLConnection
            .setDefaultHostnameVerifier(new CustomizedHostnameVerifier());
        HttpsURLConnection conn = (HttpsURLConnection) url.openConnection();
        conn.setDoOutput(true);
        conn.setUseCaches(false);
        conn.setRequestMethod("POST");
        conn.setRequestProperty("Content-Type",
            "application/x-www-form-urlencoded;charset=UTF-8");
    }
}
```

```

        conn.setRequestProperty("Content-Length",
                               Integer.toString(postData.length));
        conn.setRequestProperty("Authorization", "GoogleLogin auth="
                               + auth_token);
        OutputStream out = conn.getOutputStream();
        out.write(postData);
        out.close();
        int responseCode = conn.getResponseCode();
        return responseCode;
    }
    private static class CustomizedHostnameVerifier implements HostnameVerifier {
        public boolean verify(String hostname, SSLSocket session) {
            return true;
        }
    }
}

```

در پایان کلاسی به نام "SendMessageToDevice" که وظیفه‌ی آن منحصر ارسال پیغام به دستگاه شما می‌باشد.

```

package de.vogella.java.c2dm.server;
import java.io.IOException;
import de.vogella.java.c2dm.server.util.MessageUtil;
public class SendMessageToDevice {
    public static void main(String[] args) throws IOException {
        // "Message to your device." is the message we will send to the Android app
        int responseCode = MessageUtil.sendMessage(
                ServerConfiguration.AUTHENTICATION_TOKEN,
                ServerConfiguration.REGISTRATION_ID, "Message to your device.");
        System.out.println(responseCode);
    }
}

```

اپلیکیشن را اجرا نمایید. اپلیکیشن حاضر می‌بایست پیغامی به دستگاه شما ارسال نموده و کد بازگشتی "200" را در خروجی برگرداند. خواهید دید که در نمایشگر دستگاه یک اطلاعیه (notification) قابل مشاهده است که با کلیک بر روی آن پیغام مورد نظر نشان داده می‌شود.

## بخش پنجم

### توابع کتابخانه‌ای مربوط به تقویم / آموزش Calendar API

آموزش حاضر به شرح نحوه‌ی فراخوانی و پیاده‌سازی توابع کتابخانه‌ای مربوط به تقویم یا Calendar API در اندروید می‌پردازد. پروژه و مثال‌های مبحث حاضر در محیط برنامه نویسی

با زبان Java 1.6 نوشته شده و مبتنی بر ویرایش 4.0 سیستم عامل اندروید می باشد.

API از ویرایش 4.0 سیستم عامل اندروید در اختیار توسعه دهنگان اپلیکیشن های اندرویدی قرار گرفت.

جهت اعلان رخدادهای جدید (event) کافی است از طریق آبجکت های Intent اقدام نمایید. برای تعریف رخداد نیازی به دریافت مجوز نیست.

برای مقداردهی property ها و ویژگی های event مورد نظر لازم است از متغیر مربوط به آبجکت های Intent که برای تبادل داده های اضافی بین activity ها مورد استفاده قرار می گیرند، استفاده نمایید. اپلیکیشن شما از کاربر می پرسد آیا رخداد (event) ایجاد شود یا خیر.

به عنوان مثال، کد زیر از کاربر می پرسد آیا رخدادی با جزئیات معین ایجاد شود یا خیر.

```
// ACTION_INSERT does not work on all phones
// use Intent.ACTION_EDIT in this case
Intent intent = new Intent(Intent.ACTION_INSERT);
intent.setData(CalendarContract.Events.CONTENT_URI);
startActivity(intent);
```

در آینده مجدداً رخداد، می توانید تاریخ و زمان را نیز اضافه event در صورتی که قرار است نمایید.

```
Intent intent = new Intent(Intent.ACTION_INSERT);
intent.setType("vnd.android.cursor.item/event");
intent.putExtra(Events.TITLE, "Learn Android");
intent.putExtra(Events.EVENT_LOCATION, "Home suit home");
intent.putExtra(Events.DESCRIPTION, "Download Examples");
// Setting dates
GregorianCalendar calDate = new GregorianCalendar(2012, 10, 02);
intent.putExtra(CalendarContract.EXTRA_EVENT_BEGIN_TIME,
    calDate.getTimeInMillis());
intent.putExtra(CalendarContract.EXTRA_EVENT_END_TIME,
    calDate.getTimeInMillis());
// make it a full day event
intent.putExtra(CalendarContract.EXTRA_EVENT_ALL_DAY, true);
// make it a recurring Event
intent.putExtra(Events.RRULE, "FREQ=WEEKLY;COUNT=11;WKST=SU;BYDAY=TU,TH");
```

```
// Making it private and shown as busy
intent.putExtra(Events.ACCESS_LEVEL, Events.ACCESS_PRIVATE);
intent.putExtra(Events.AVAILABILITY, Events.AVAILABILITY_BUSY);
```



966

آدرس آموزشگاه : تهران - خیابان شریعتی - بالاتر از خیابان ملک - جنب بانک صادرات - پلاک 651 طبقه دوم - واحد 7

88146323 - 88446780 - 88146330