عنوان: **توزیع کلاس‌های اندرویدی با استفاده از Gradle قسمت اول**
نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۴/۰۸/۰۹ ۲۰:۵
آدرس: www.dotnettips.info
گروه‌ها: Android, Java, Gradle

یکی از روش‌هایی که امروزه مورد استقبال برنامه نویسان اندروید و جاوا قرار گرفته‌است، استفاده از یک سیستم DSL به نام Gradle (+) است. ابتدا در سیستم‌های Apache Ant (+) و Maven (+) مورد استفاده قرار می‌گرفت، ولی با جمع کردن نقاط ضعف آن دو سیستم، و رفع عیوب آن‌ها و افزودن مزیت‌های جدید، Gradle ایجاد شد. یکی از استفاده‌هایی که به شدت مورد استفاده‌ی برنامه نویسان اندروید قرار می‌گیرد، استفاده از یک سیستم توزیع برای کلاس‌های اندرویدی است. اگر امروزه به خیلی از سورس‌های قرار گرفته بر روی گیت هاب، نگاه کنید، به غیر از روش افزودن آن سورس به پروژه به عنوان ماژول، روش دیگری نیز وجود دارد که آن، استفاده از سیستم توزیع Gradle است. استفاده از این روش محبوبیت زیادی دارد و بسیار هم راحت‌تر است از افزودن یک سورس به پروژه.

برای افزودن یک ماژول به پروژه از طریق گریدل، به صورت زیر اقدام می‌کنیم:
هر ماژول شامل یک فایل به نام build.gradle است که تنظیمات سطح آن ماژول را به عهده دارد و پروژه نیز یک build.gradle دارد که تنظیمات آن در سطح پروژه صورت می‌گیرد. برای اقزودن سورس در سطح یک ماژول لازم است که تعدادی خط کد را که معرف و آدرس آن سورس را دارد، به فایل build.gradle اضافه کنیم. به عنوان مثال برای سورس Active Android را که یک ORM عالی در سطح اندروید به شمار می‌آید، به ماژولمان اضافه می‌کنیم:

```
dependencies {
    compile 'com.michaelpardo:activeandroid:3.1.0-SNAPSHOT'
}
```

ولی یک سوال پدید می‌آید که این خط کوتاه که تنها شامل نام و نسخه‌ی کتابخانه است، چگونه می‌تواند آدرس آن سورس را به دست بیاورد؟ در این نوشتار قصد داریم این مساله را بررسی کرده و بدانیم که این سورس‌ها چگونه به این سیستم توزیع اضافه شده‌اند.

سوال : اندروید استودیو، کتابخانه‌های اندرویدی را از کجا دانلود می‌کند؟

Apache Maven یک سیستم آزاد است که برای توزیع کتابخانه‌ها مورد استفاده قرار می‌گیرد. سرورهای این سیستم شامل یک مخزن maven هستند که کتابخانه‌ها در آن قرار می‌گیرند و شناسه‌ی دسترسی به آن کتابخانه از طریق همان شناسه‌ای است که شما در build.gradle تعریف می‌کنید. به طور عادی دو سرور استاندارد برای اینکار وجود دارند که یکی از آن‌ها jcenter و دیگری mavenCnteral است. البته سرورهای دیگری نیز وجود دارند، یا اینکه حتی خودتان هم می‌توانید میزبانی را به عهده بگیرید و یا بعضی از شرکت‌ها برای خود مخزنی جداگانه دارند.

JCenter

این سرور که یک مخزن maven است، توسط Bintray میزبانی می‌شود که می‌توانید آن را در این آدرس بیابید. برای اینکه شناسه‌های gradle مربوط به این سرور در اندروید استودیو دانلود شود، نیاز است خط زیر را به build.gradle سطح پروژه اضافه کنید:

```
allprojects {
    repositories {
        jcenter()
    }
}
```

MavenCentral

این مخزن توسط Sonatype.org میزبانی می‌شود که کل مخزن آن را می‌توانید در این آدرس بیابید. برای دسترسی به مخازن این

سرور نیاز است خطوط زیر را به gradle سطح پروژه اضافه کنید:

```
allprojects {
    repositories {
        mavenCentral()
    }
}
```

موقعی که شما شناسه‌ی گریدل را اضافه می‌کنید، حتما باید دقت کنید مخزن آن کجا قرار گرفته است؟ آیا در یکی از آدرس‌های بالاست یا حتی می‌تواند در هر دو آدرس بالا قرار گرفته باشد؛ یا مخزنی غیر از مخازن بالاست.

به عنوان مثال Twitter's Fabric.io خودش کتابخانه‌ی خودش را میزبانی می‌کند و مخزن آن در این آدرس https://maven.fabric.io/public قرار گرفته است و برای افزودن این کتابخانه به پروژه نیاز است مسیر زیر طی شود:

```
//project build.gradle
repositories {
    maven { url 'https://maven.fabric.io/public' }
}

//module build.gradle
dependencies {
    compile 'com.crashlytics.sdk.android:crashlytics:2.2.4@aar'
}
```

سوال : کدامیک از مخازن بالا را انتخاب کنیم؟

اینکه کدامیک را انتخاب کنیم مساله ای است که به خودتان مرتبط است و هر دو برای یک هدف ایجاد شده‌اند و آن میزبانی کتابخانه‌های جاوا و اندرویدی است. بسیاری از توسعه دهندگان از هر دو استفاده می‌کنند؛ ولی بعضی‌ها هم تنها یکی از آن دو را بر می‌گزینند. اندروید استودیو در نسخه‌های اولیه‌ی خود به طور پیش فرض mavencenteral را صدا میزد و به طور پیش فرض در buil.gradle پروژه، آن را معرفی کرده بود. ولی مساله ای که در این بین بود، این بود که این مخزن چندان به مذاق توسعه دهندگان خوش نمی‌آمد و کمی کار با آن دشوار و سخت بود. لذا تیم اندروید بنا به دلایلی مثل آن و موارد امنیتی و ... و اینکه توسعه دهندگان بیرونی بیشتر از jcenter استفاده می‌کردند، آن هم به سمت jcenter رفتند که در ورژن‌های فعلی اندروید استودیو می‌توانید ببینید که کتابخانه‌ی پیش فرض تغییر یافته است و ()jcenter به جای ()mavencentral صدا زده می‌شود.

دلایل مهاجرت از mavencentral به jcenter:

سیستم jcenter از طریق یک CDN عمل می‌کند که در این صورت می‌تواند تجربه‌ی خوبی از سرعت بهتر را برای توسعه دهندگان به همراه داشته باشد.
کتابخانه‌های jcenter بسیار بیشتر از mavencentral هستند؛ تا جایی که می‌توان گفت اکثر کتابخانه‌هایی که روی mavencenteral پیدا می‌شوند، روی jcenter هم هست و jcenter بزرگترین مخزن به شمار می‌آید.
آپلود کتابخانه بر روی jcenter بسیار راحت‌تر است و نیاز به کار پیچیده‌ای ندارد.
در این نوشتار سعی داریم ابتدا کتابخانه‌ی AndroidBreadCrumb را بر روی jcenter آپلود کنیم و سپس با استفاده از روش آسانتری آن را به سمت mavenCentral بفرستیم.

بررسی قسمت‌های یک شناسه Gradle

هر شناسه شامل سه قسمت می‌شود:

```
GROUP_ID:ARTIFACT_ID:VERSION
```

قسمت اول نام پکیج است که به آن Group_ID می‌گویند و می‌تواند خانواده‌ای از یک پکیج را نیز مشخص کند. سپس قسمت Artifact، نامی است که بر روی پروژه‌ی خود گذاشته‌اید و سپس ورژن است که در قالب x.y.z معرفی می‌شود و در صورت

اختیار می‌توانید عباراتی مثل beta- و snapshot- را هم داشته باشید.
کتابخانه‌های زیر، از یک خانواده هستند که به راحتی می‌توانید آن‌ها را از هم تشخیص دهید:

```
dependencies {
  compile 'com.squareup:otto:1.3.7'
  compile 'com.squareup.picasso:picasso:2.5.2'
  compile 'com.squareup.okhttp:okhttp:2.4.0'
  compile 'com.squareup.retrofit:retrofit:1.9.0'
}
```

پس نحوه‌ی دریافت کتابخانه‌ها به این شکل است که ابتدا اندروید استودیو به ترتیب مخازن معرفی شده‌ی در سطح پروژه را چک می‌کند و از طریق شناسه‌ی آن‌ها بررسی می‌کند که آیا این کتابخانه اینجا موجود است، یا خیر و اگر موجود بود آن را دانلود می‌کند.

شناخت فایل‌های AAR

همانطور که می‌دانید فرمت فایل‌های بایت کدی جاوا JAR می‌باشد که هم توسط جاوا و هم اندروید پشتیبانی می‌شود. ولی در صورتیکه کلاس شما یک پروژه‌ی اندرویدی باشد، نمی‌توانید آن را در قالب یک فایل JAR منتشر کنید. چرا که که کلاس اندرویدی می‌تواند شامل فایل مانیفست، منابع و ... باشد که در فایل JAR جایی برای آن‌ها مهیا نشده است. به همین علت فایل‌های نوع AAR برای اینکار مهیا شده‌اند که این فایل در واقع یک فایل زیپ است که محتویات مورد نظر داخل آن قرار گرفته است و یکی از آن فایل Classes.jar برای کدهاست و مابقی آن به شرح زیر است:

```
- /AndroidManifest.xml (الزامی)
- /classes.jar (الزامی )
- /res/ (الزامی )
- /R.txt (الزامی )
- /assets/ (اختیاری)
- /libs/*.jar (اختیاری )
- /jni/<abi>/*.so (اختیاری )
- /proguard.txt (اختیاری )
- /lint.jar (اختیاری )
```

در مقاله‌ی بعدی کار را با jcenter آغاز می‌کنیم.

نظرات خوانندگان

نویسنده: مرتضی حاتمی
تاریخ: ۱۳۹۴/۰۸/۰۹ ۲۲:۵۸

سلام
اگر میشه کمی در مورد نسخه beta- و snapshot- بیشتر توضیح دهید.

نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۴/۰۸/۱۰ ۲:۱۹

snaphot بدین معنی است که این نسخه در حال حاضر در حال توسعه است و به نسخه حقیقی آن نرسیده است و ممکن است نسخه ای که دیروز از گریدل گرفتید با نسخه ای که امروز از گریدل دانلود می‌کنید متفاوت باشه و در هر مرحله زمانی به روزرسانی شود و بیشتر جهت بررسی آن نسخه ارائه گردیده است و این احتمال میرود هنوز به مرحله پایداری نرسیده است.
در حالی که نسخه بتا یک نسخه تست شده در محیط توسعه است و یک سری امکاناتی را ارئه داده است و جهت تست و رفع عیب در محیط عملیاتی ارائه گشته است تا نسخه بتای بعد یا نهایی و ... ارائه گردد